AD-A081 836 AERONAUTICAL RESEARCH LABS MELBOURNE (AUSTRALIA) F/G 9/2
COMPUTER SYNTHESIS OF FLIGHT SIMULATION VISUALS.(U)
FEB 79 J SANDOR
UNCLASSIFIED ARL/SYS-NOTE-61 NL

1 OF 1
AD
A081836

END
DATE
FILMED
4-80
DTIC

ARL-SYS-NOTE-61 ✓

AR-001-593

LEVEL

ADA081836

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

AERONAUTICAL RESEARCH LABORATORIES

MELBOURNE, VICTORIA

SYSTEMS NOTE 61

COMPUTER SYNTHESIS OF FLIGHT SIMULATION VISUALS

by

JOHN SANDOR

Approved for Public Release

DTIC ELECTE MAR 17 1980 D A

DDC FILE COPY

© COMMONWEALTH OF AUSTRALIA 1979

COPY No 14

FEBRUARY 1979

80 3 14 71

APPROVED
FOR PUBLIC RELEASE

THE UNITED STATES NATIONAL
TECHNICAL INFORMATION SERVICE
IS AUTHORISED TO
REPRODUCE AND SELL THIS REPORT

AR-001-593

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
AERONAUTICAL RESEARCH LABORATORIES

14 ARL SYS-1 TF-61

9 SYSTEMS NOTE 61

COMPUTER SYNTHESIS OF FLIGHT SIMULATION VISUALS

by

10 JOHN SANDOR

11 Feb 77 12 41

A

SUMMARY

A review is presented of the principles of computer synthesis of external world visuals for raster displays. The areas of environment description, edge and polygon clipping, hidden surface removal, shading and image aberrations are discussed. Particular attention is given to hierarchical environment structure, image coherence, shading functions and area-averaging edge smoothing techniques.

POSTAL ADDRESS: Chief Superintendent, Aeronautical Research Laboratories, Box 4331, P.O., Melbourne, Victoria, 3001, Australia.

DOCUMENT CONTROL DATA SHEET

Security classification of this page: Unclassified

1. Document Numbers (a) AR Number: AR-001-593 (b) Document Series and Number: Systems Note 61 (c) Report Number: ARL-Sys-Note-61	2. Security classification (a) Complete document: Unclassified (b) Title in isolation: Unclassified (c) Summary in isolation: Unclassified
3. Title: COMPUTER SYNTHESIS OF FLIGHT SIMULATION VISUALS	
4. Personal Author(s): John Sandor	5. Document Date: February, 1979 6. Type of Report and Period Covered:
7. Corporate Author(s): Aeronautical Research Laboratories 9. Cost Code: .7/2260	8. Reference Numbers: (a) Task: (b) Sponsoring Agency:
10. Imprint (Publishing establishment): Aeronautical Research Laboratories, Melbourne	11. Computer Program(s) (Title(s) and language(s)):
12. Release Limitations (of the document): Approved for public release	

12-0. Overseas:	N.O.		P.R.	1	A		B		C		D		E	

13. Announcement Limitations (of the information on this page): No limitation

14. Descriptors:		15. Cosati Codes:
Computerized simulation	Flight simulation	0902
Graphic methods	Computer graphics	1201
		1402

16. *ABSTRACT*

A review is presented of the principles of computer synthesis of external world visuals for raster displays. The areas of environment description, edge and polygon clipping, hidden surface removal, shading and image aberrations are discussed. Particular attention is given to hierarchical environment structure, image coherence, shading functions and area-averaging edge smoothing techniques.

CONTENTS

NOTATION

R^3	Euclidean 3-space
$a, b, \ldots$	denote points in R^3
a_x, a_y, a_z	co-ordinates of the point a in R^3
$\underline{a}, \underline{b}, \ldots$	denote vectors from the origin to $a, b, \ldots$ in R^3
$\overline{ab}$	is the directed line segment from a to b
$\underline{n}$	is the outward normal direction to a plane
$\underline{a} \cdot \underline{b}$	dot product between $\underline{a}$ and $\underline{b}$
$\underline{a} \times \underline{b}$	cross product between $\underline{a}$ and $\underline{b}$
$\underline{a}^T$	transpose of the vector $\underline{a}$
$\lVert(\cdot)\rVert$	Euclidean norm
$\lvert\cdot\rvert$	absolute value
$OXYZ$	environment frame of reference
$O'X'Y'Z'$	observer frame of reference
$\underline{a}', \underline{b}', \ldots$	denotes vectors in $O'X'Y'Z'$
n'	is the outward normal direction in $O'X'Y'Z'$
$O_SX_SY_S$	view plane co-ordinate system
Π	view plane
$\theta_1, \theta_2, \theta_3$	Euler angles
T	transformation matrix
t_{ij}	(i, j)-th element of T

1. INTRODUCTION

The use of manned flight simulators in military aviation is now well established in the USA, UK and other NATO countries. Currently consideration is being given to the establishment of a DSTO research simulation facility to complement Service training simulators and detailed mathematical modelling studies. As part of this work investigations are being made into key simulation methodologies and this paper deals with the synthesis of the external visual environment.

The main approaches to flight simulation visuals are:

(i) *Shadow graph.* This comprises a transparency illuminated by a point light source to give an image on a curved screen. The field of view can be quite large, resolution is normally good whilst changes in orientation are simulated faithfully. However the range of light and terrain simulated is restricted by the limited visual content and total possible movement of the transparency with respect to the light source. Hence the method has most utility in rendering earth/sky backgrounds in combat simulation.

(ii) *Anamorphic Motion Picture.* A cine film is made along a specified flight path over the area of interest, and is viewed by projection with optical correction used to simulate motion with respect to the flight path. The approach has some desirable features: cine film resolution is good, wide field of view is possible and three-dimensional objects appear in correct perspective. The method of simulating movement of the viewpoint away from the viewpoint of the camera originally used to make the film consists of distorting the image by use of variable anamorphic lenses which produce magnification that is controllable in magnitude and direction across the field of view. While this method gives good results for objects in the ground plane, objects standing up above the ground are compressed/expanded and lean to the left or right,[26] particularly for large excursions from the desired flight path. The latter phenomenon, together with the limited range of visual environments possible, restrict the applicability of the approach to take-off and landing simulation.

(iii) *Scanned Physical Model.* In this method, a contoured physical model is scanned by a television camera with subsequent reproduction on an appropriate display device. Careful modelling and high level of detail ensure excellent rendition of cultural features. Perspective changes are achieved by vertical and lateral motion of the television camera.

A modified version of this approach has been proposed in Reference 26. Instead of a television camera, the model is scanned, in television raster format, by a laser beam. Arrays of photomultipliers collect the reflected light and modulate a scanned laser projector. Perspective and motion are synthesized by the position of the exit pupil of the laser and the orientation of the raster.

There are two major limitations of scale model based visuals. First, the relatively large amount of detail present in the model precludes frequent changes in the environment definition, and second, the restricted geographical area represented, together with opto-mechanical limitations, restricts the range of simulated manoeuvres. Nevertheless, both civilian and military pilot training have successfully employed scale models for limited terrain simulations such as take-off and landing.

(iv) *Raster based Computer Generated Imagery* (CGI). In this approach images are generated by computation based on numerically stored environment data representing polyhedral models of environment objects in a Cartesian co-ordinate system. The scene generation process consists of two phases: the first involves mapping of the three-dimensional environment onto a two-dimensional view plane, at the same time eliminating out of view objects from further processing. The second phase involves the solution of the hidden surface problem, i.e. determining which object surface is visible at each picture element.

The CGI approach to synthesis of visuals is particularly suited to research orientated simulation. In comparison to scale models it offers increased versatility: there are no serious limitations to the geographical area modelled, the inclusion or exclusion of specific visual cues, nor the range of manoeuvres possible. Although the rendering of fine detail is at present limited in CGI, current trends in digital processing point toward rapidly improving capability.

This report reviews, at the conceptual level, the present methodology for raster CGI and delineates the significant aspects. Geometrical considerations pertaining to environment modelling and transformations are discussed in Sections 2 and 3 whilst clipping, the technique of distinguishing between in-view and out-of-view objects is considered in Section 4. In Section 5 a number of hidden surface algorithms are examined with particular reference to sorting and image coherence utilization. Image realism, which here will be taken to mean those characteristics of an image whose presence or absence attenuates realism, is considered in Section 6. Since this area is most prone to subjective judgment, attention is restricted to the areas of shading and raster quantization effects. In particular, methods are discussed for the synthesis of distance and horizon fading, smoothing of curved surfaces approximated by polygons and area averaged edge smoothing techniques.

Details pertaining to system specific aspects of CGI, such as computer and display hardware will be treated in a subsequent report.

2. GEOMETRICAL CONSIDERATIONS

2.1 The Environment

The visual environment for flight simulation comprises three classes of objects:

I a planar surface representing a local "flat earth", objects on the earth's surface including terrain, and objects above the earth such as aircraft and clouds;

II a diffuse atmosphere with the property that the surface colours of objects appear to desaturate at a large distance from the observer;

III the sky, which is the upper part of the atmosphere (and beyond) as seen from the surface of the earth.

The modelling of objects of class I will now be considered whilst the representation of classes II and III is deferred to Section 6.

It will be assumed that there exists a Cartesian co-ordinate system, with origin located within the region of interest such that each object of class I can be located and described in this co-ordinate system, with objects (including the terrain) being modelled as convex polyhedra. Henceforth when reference is made to an object it is to be assumed that the polyhedral model of the object is referred. Although the assumption of polyhedral models may seem unduly restrictive, it does result in clipping and hidden surface algorithms involving only linear equations. Moreover it will be shown in Section 6 that the smooth appearance of a curved object, modelled as a polyhedron, can be restored by shading.

Thus each object can be described, in the aforementioned co-ordinate system, by the location of its vertices and the relationship between the vertices and the faces of the object.

In addition to defining single objects, it is sometimes advantageous to classify the environment into clusters. A *cluster* is a collection of objects (or faces of an object) which can be treated as a single entity with respect to a particular property. For example, a cluster may consist of all the faces belonging to a single object, or it may be defined by regional extent. Further, two clusters are said to be *linearly separable* if a plane separates them. An alternative descriptor is a quadratically separable cluster; that is, one which can be completely enclosed by a sphere. It will be shown that the centre and radius of such a sphere are useful parameters for a particular form of clipping.

2.2 Viewing Geometry

In this section some useful geometric relationships are reviewed and the geometry of viewing a polyhedral object in a scene is defined.

Let a, b, c be three points defining a plane in R^3. Then the outward normal direction is given by:

$$\underline{n} = \overline{ac} \times \overline{bc} = \begin{vmatrix} i & j & k \\ c_x - a_x & c_y - a_y & c_z - a_z \\ c_x - b_x & c_y - b_y & c_z - b_z \end{vmatrix}$$

If a is a point on a plane with outward normal $\underline{n}$, then the equation of the plane is

$$\underline{n} \cdot (\underline{x} - \underline{a}) = 0; \ \underline{x}^T = (x, y, z)$$

Let $v_i(x_i, y_i, z_i)$, $i = 1, \ldots, n$ be the n vertices of a polygon. Then the *centroid* of the polygon is

$$c = \frac{1}{n} \sum_{i=1}^{n} v_i$$

The geometry of viewing an object in a scene is illustrated in Fig. 2.1.

The frame $OXYZ$ represents the environment co-ordinate system whilst $O'X'Y'Z'$ is the frame attached to the observer. The location of the vertices of an object and the origin of the observer frame are specified in environment co-ordinates. The viewing plane, Π is parallel to the $O'X'Y'$ plane and the view frame $O_SX_SY_S$ is in the usual sense as viewed by the observer.

2.3 Data Base Preparation

It is clear from the preceding remarks that the data required to numerically describe an object, in the sense of spatial extent, are the co-ordinates of the object's vertices. However, in a number of image synthesis algorithms (e.g. clipping and hidden surface) the quantities of interest are not vertices *per se* but edges and polygons. This implies that the vertices defining the environment have to be indexed and stored in an order compatible with the algorithms such that edges and faces can be readily determined.

There are a number of ways one may index and store the vertex data and it is instructive to consider the following two approaches.

I. Fixed Sequence Non-repeated Vertices

In this method the vertices are indexed and stored in a fixed sequence corresponding to an *a priori* defined set of relationships between vertices, edges and faces. As an example consider the oblong of Fig. 2.2 and let the following table define the relationship between vertices and faces stored in the computer. (Note that the sequence of vertices in the middle column is in the "right hand screw sense" with respect to outward surface normal.)

Face No.	Vertices	Adjacent Faces
f_1	$v_1\ v_2\ v_3\ v_4$	$f_2\ f_4\ f_5\ f_6$
f_2	$v_1\ v_4\ v_5\ v_6$	$f_1\ f_3\ f_5\ f_6$
f_3	$v_6\ v_5\ v_8\ v_7$	$f_2\ f_4\ f_5\ f_6$
f_4	$v_3\ v_2\ v_7\ v_8$	$f_1\ f_3\ f_5\ f_6$
f_5	$v_3\ v_4\ v_5\ v_8$	$f_1\ f_2\ f_3\ f_4$
f_6	$v_1\ v_2\ v_7\ v_6$	$f_1\ f_2\ f_3\ f_4$

Note that the indexing of the figure in Fig. 2.2 corresponds to the above rules of association between vertices and faces. If however the object is wrongly indexed the resulting synthesized image will have little resemblance to the true object. In addition, it is evident that there must be a set of association rules, similar to the above, for each class of polyhedra (such as tetrahedrons, pyramids and octagons).

Although the present approach is cumbersome and requires careful indexing of environment vertices it requires minimal memory since each vertex is only entered once; this implies that the approach is efficient with respect to those algorithms that involve processing each vertex independently. This is in contrast to the method discussed next.

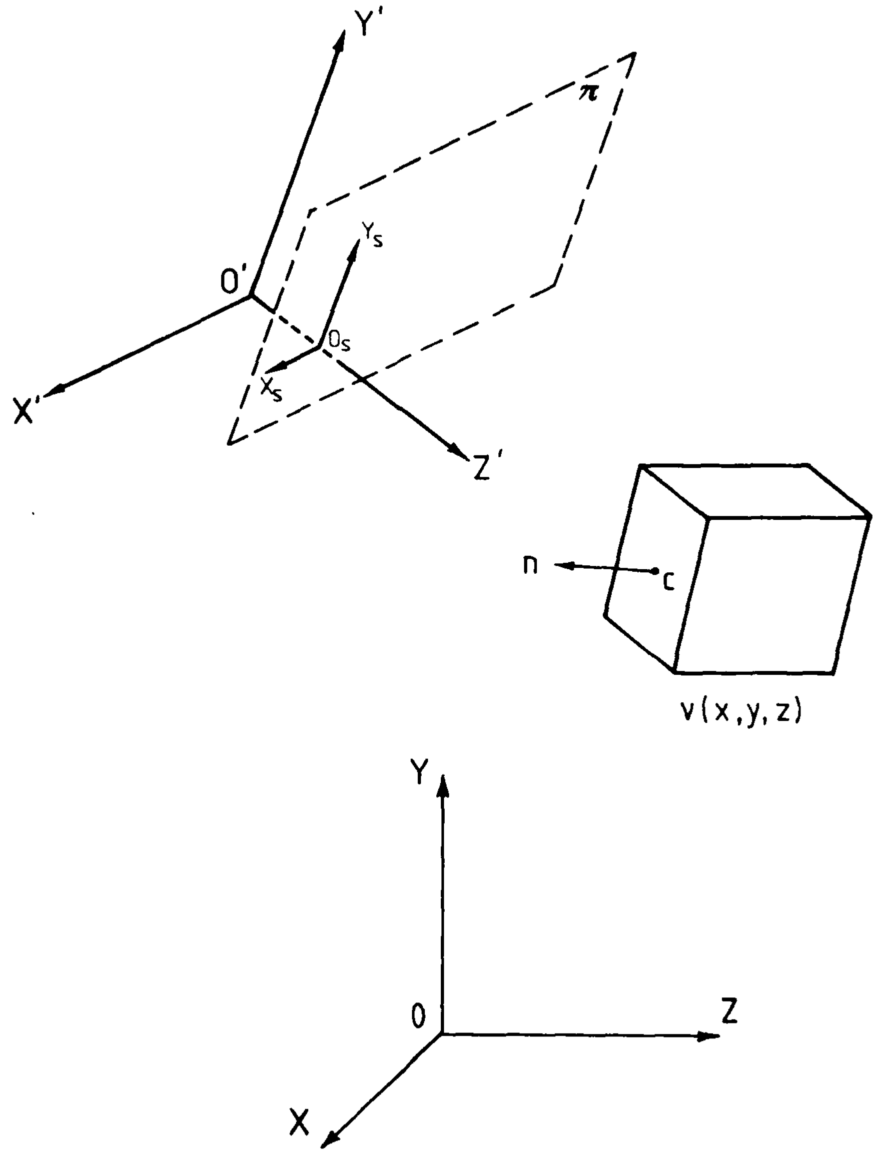


FIG. 2.1 VIEW GEOMETRY

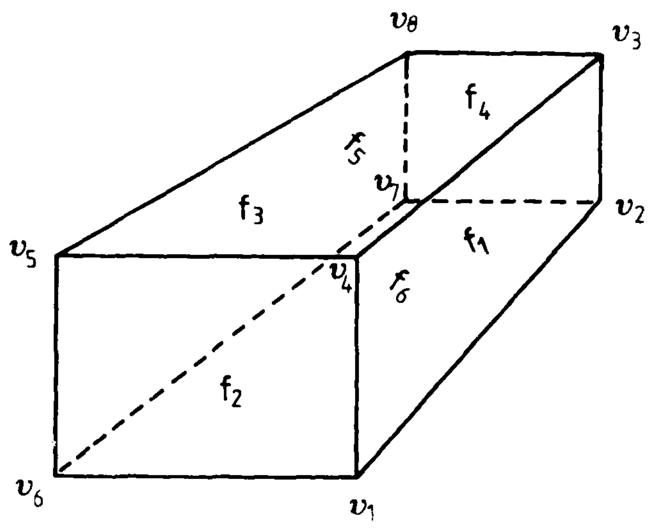


FIG. 2.2

II. Polygon Based Vertex Sequence

In this approach objects in the scene are partitioned into faces (which by assumption are polygonal), the vertices of each polygon are indexed in the right hand screw sense corresponding to the outward surface normal. The implication is, as far as numerical data description is concerned, that object faces rather than objects are to be treated as the quantities of interest. Thus for the oblong of Fig. 2.2 the following two vertex sequences are admissible:

$$v_6\,v_1\,v_4\,v_5 \quad v_1\,v_2\,v_3\,v_4 \quad v_2\,v_7\,v_8\,v_3 \quad v_7\,v_6\,v_5\,v_8 \quad v_3\,v_8\,v_5\,v_4 \quad v_1\,v_6\,v_7\,v_2$$

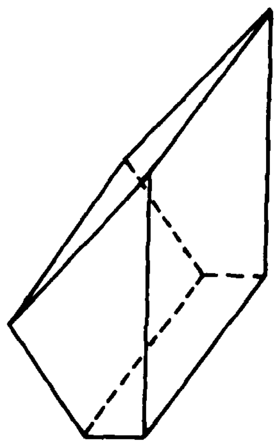
or

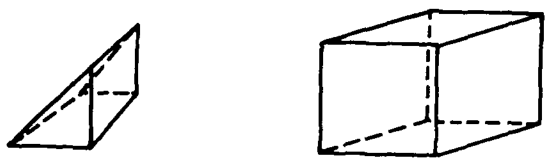
$$v_1\,v_2\,v_3\,v_4 \quad v_6\,v_1\,v_4\,v_5 \quad v_2\,v_7\,v_8\,v_3 \quad v_7\,v_6\,v_5\,v_8 \quad v_7\,v_8\,v_5\,v_4 \quad v_1\,v_6\,v_7\,v_2$$

The approach proves convenient from the point of view of polygon based algorithms although the repetition of each vertex in the sequence requires increased storage capacity. Moreover this implies that the approach is inefficient with respect to vertex and edge based algorithms (such as co-ordinate transformations and clipping). In the example considered above each vertex appears three times whilst each edge appears twice.

A conceptually more appealing approach than the preceding two methods is to represent the environment objects not in terms of co-ordinates of vertices but in terms of position and dimension of "higher level" primitives such as cubes, oblongs, tetrahedrons and so on. Then each time a primitive appeared in the environment, an algorithm, corresponding to the primitive, would be invoked to generate its vertices, edges or faces. The efficiency of the approach is dependent on the careful characterization of objects in terms of the primitives in order to minimize the number of superfluous edges and vertices that have to be removed from the resultant image. For example consider the object in Fig. 2.3(*a*) and assume it is to be represented in terms of the primitives of Fig. 2.3(*b*). One representation (in exploded view) is shown in Fig. 2.3(*c*) and it is clear that the shaded surfaces are redundant. A more economic representation is that of Fig. 2.3(*d*).

An extension of the above approach is to have primitives that are actually objects such as aircraft, trees, or mountains, and use them as the basis for environment representation. The key difficulty here, particularly with regard to objects such as mountains, is the numerical characterization of descriptive qualities. For example, if a mountain is to be represented it is desirable that in addition to geographical position and extent, attributes such as "ruggedness" be also numerically characterized. Methodology to handle this problem has yet to be forthcoming.


(a)


(b)

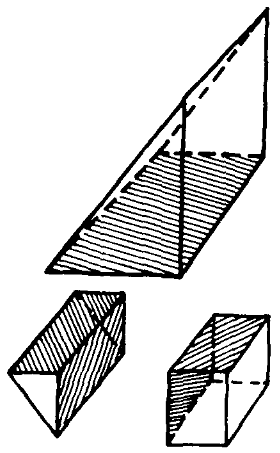
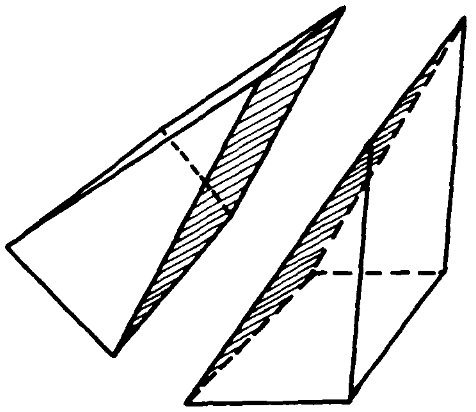

(c)

FIG. 2.3



(d)

FIG. 2.3

3. TRANSFORMATIONS

For a scene, described in environment co-ordinate system $OXYZ$, many images can be generated depending on the position from which the scene is viewed. To generate a specific image—namely the image of the scene as viewed by the observer, the objects comprising the scene have to be described in terms of the co-ordinate system $O_SX_SY_S$. The machinery necessary to achieve this is now developed.

3.1 Translation and Rotation

Let the position of O' in $OXYZ$, be specified by the triple (O_x', O_y', O_z') and let the orientation of $O'X'Y'Z'$ with respect to $OXYZ$ be specified by the angles θ_1, θ_2 and θ_3 (Fig. 3.1).

θ_1 is the angle of rotation about OY axis;
θ_2 is the angle of rotation about OX axis;
θ_3 is the angle of rotation about OZ axis.

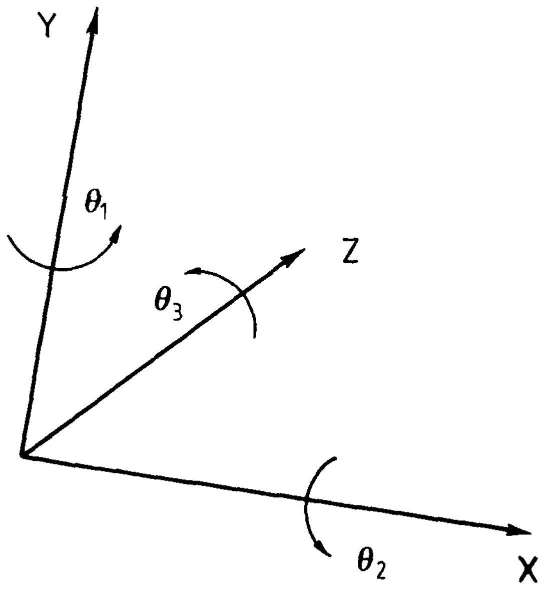


FIG. 3.1

Then any point $P(x, y, z)$ in the frame $OXYZ$ has co-ordinates (x', y', z') in the observer frame $O'X'Y'Z'$ such that[4]

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_2 & \sin\theta_2 \\ 0 & -\sin\theta_2 & \cos\theta_2 \end{pmatrix} \begin{pmatrix} \cos\theta_1 & 0 & -\sin\theta_1 \\ 0 & 1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} x-O_x' \\ y-O_y' \\ z-O_z' \end{pmatrix} \quad (3.1.1.)$$

Note that since translation and rotation are non-commutative operations, the above expression represents the following sequence:

(*a*) translation;
(*b*) rotation about OY axis;
(*c*) rotation about OX axis;
(*d*) rotation about OZ axis.

Relationship (3.1.1.) is more conveniently expressed in matrix form if each point (x, y, z) is represented in homogeneous co-ordinates[5] by the 4-tuple $(x, y, z, 1)$. That is:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \begin{array}{c} \sin\theta_3 \sin\theta_2 \sin\theta_1 + \\ + \cos\theta_3 \cos\theta_1 \end{array} & \sin\theta_3 \cos\theta_2 & \begin{array}{c} \sin\theta_3 \cos\theta_1 \sin\theta_2 - \\ - \cos\theta_3 \sin\theta_1 \end{array} & t_{14} \\ \begin{array}{c} \cos\theta_3 \sin\theta_2 \sin\theta_1 - \\ - \cos\theta_1 \sin\theta_3 \end{array} & \cos\theta_3 \cos\theta_2 & \begin{array}{c} \cos\theta_3 \cos\theta_1 \sin\theta_2 + \\ + \sin\theta_3 \sin\theta_1 \end{array} & t_{24} \\ \sin\theta_1 \cos\theta_2 & -\sin\theta_2 & \cos\theta_2 \cos\theta_1 & t_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (3.1.2)$$

$$= T \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

where

$$t_{14} = -O_x'[\cos\theta_3 \cos\theta_1 + \sin\theta_3 \sin\theta_2 \sin\theta_1] - O_y' \sin\theta_3 \cos\theta_2 - \\ - O_z'[\sin\theta_3 \cos\theta_1 \sin\theta_2 - \cos\theta_3 \sin\theta_1]$$

$$t_{24} = O_x'[\cos\theta_1 \sin\theta_3 - \cos\theta_3 \sin\theta_2 \sin\theta_1] - O_y' \cos\theta_3 \cos\theta_2 - \\ - O_z'[\sin\theta_3 \sin\theta_1 + \cos\theta_3 \cos\theta_1 \sin\theta_2]$$

$$t_{34} = -O_x' \sin\theta_1 \cos\theta_2 + O_y' \sin\theta_2 - O_z' \cos\theta_2 \cos\theta_1 .$$

Such an apparent 33% increase in dimension in the representation of a point leads to two ameliorating circumstances. Firstly an arbitrary scale factor can be introduced that allows the use of fixed point arithmetic, which is particularly effective in matrix arithmetic routines. Secondly, the projective geometry of homogeneous co-ordinates allows for the concatenation of transformations.

3.2 Perspective Transformation

The planar projection that represents real objects, as viewed by the naked eye through a homogeneous medium is perspective. The essential features of the perspective transformation are:

(i) The x_s and y_s co-ordinates of the perspective view are obtained by dividing the x' and y' observer co-ordinates by the distance from the observer forward to the object, z'.
(ii) The "perspective depth", which preserves the straightness of lines, flatness of planes and the depth ordering, must also be computed.

For the geometry depicted in Fig. 3.2 the perspective transformation expressed in the observer co-ordinate system is:[1]

$$\begin{aligned} x_s &= x'(d/z') \\ y_s &= y'(d/z') \qquad (3.2.1.) \\ z_s &= (z' - d)(d/z') \end{aligned}$$

where d is the distance of the viewing plane from the observer. Although the perspective transformation does not preserve z' depth, it does preserve depth relationships.

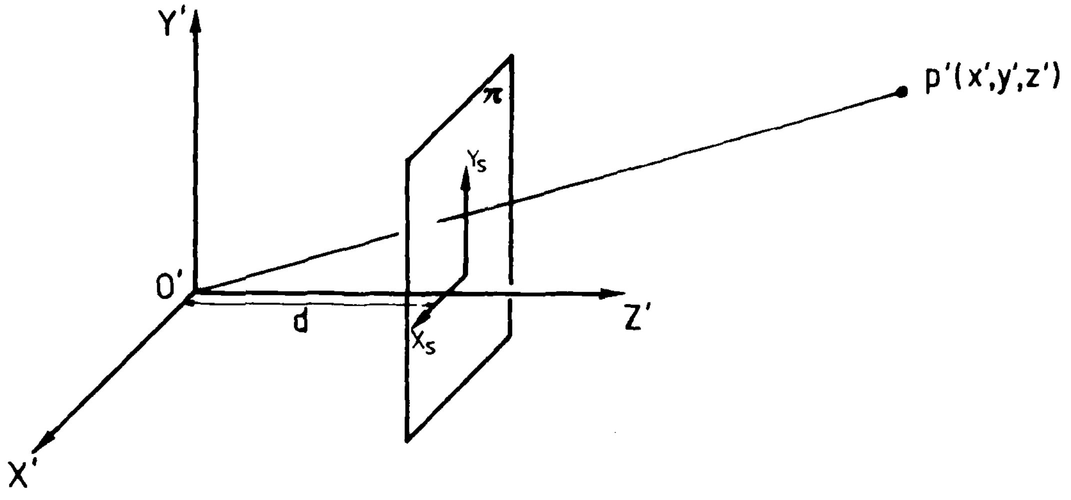


FIG. 3.2

4. CLIPPING

The process of clipping involves the sorting of all elements of a scene such that vertices and edges that are: (i) behind the viewing plane, and (ii) do not project within the boundaries of the display screen, are eliminated whilst edges that only partially project onto the display screen are truncated at the boundaries. Although this procedure is relatively simple to implement, the computational load for an environment of high complexity becomes inordinate for real time image generation.

4.1 Cluster Clipping

One way to speed up the clipping process is to characterize the environment in a structural form—typically clusters. Recall that a cluster is any object (or collection of objects) enclosed in a bounding volume. Then, as a first step in clipping, those clusters whose bounding volumes produce images outside the display screen can be eliminated from further processing whilst those clusters whose images are within the display screen can by-pass the subsequent stages of clipping.

To illustrate cluster clipping, consider a cluster whose bounding volume is a sphere with centre at (c_x, c_y, c_z) and radius r. Applying the transformation of Section 3 to this sphere, it can be readily shown that in the $O'X'Y'Z'$ frame the surface is again a sphere, with centre $(c_{x'}, c_{y'}, c_{z'})$ and radius r, where:

$$\begin{pmatrix} c_{x'} \\ c_{y'} \\ c_{z'} \\ 1 \end{pmatrix} = T \begin{pmatrix} c_x \\ c_y \\ c_z \\ 1 \end{pmatrix}$$

The projection of this sphere onto the Π plane is an ellipse. Projecting this ellipse onto the $X'O'Z'$ and $Y'O'Z'$ planes, a bounding rectangle for the image of a cluster is defined as follows:

With reference to Fig. 4.1(a), the length of the projection in the $X'O'Z'$ plane is

$$l_{x'z'} = b'' - a''$$

where

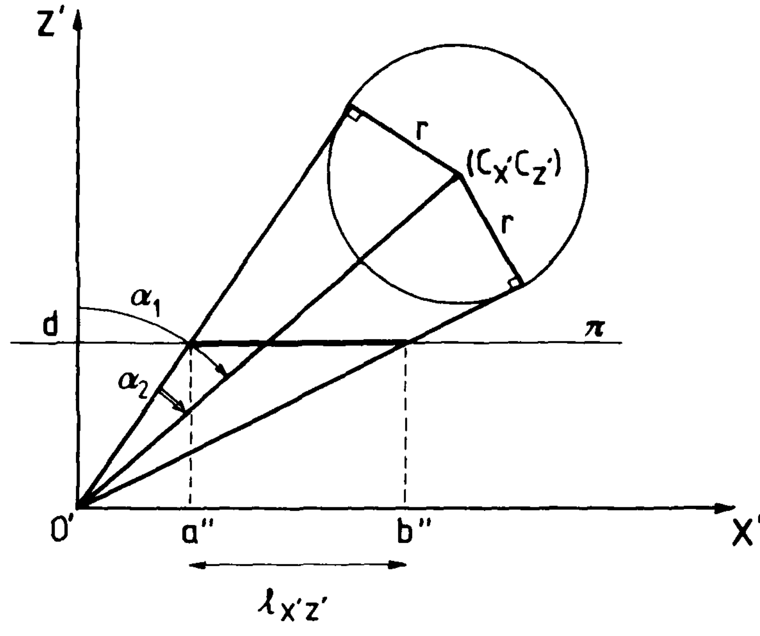
$$a'' = d \tan(\alpha_1 - \alpha_2)$$

$$b'' = d \tan(\alpha_1 + \alpha_2)$$

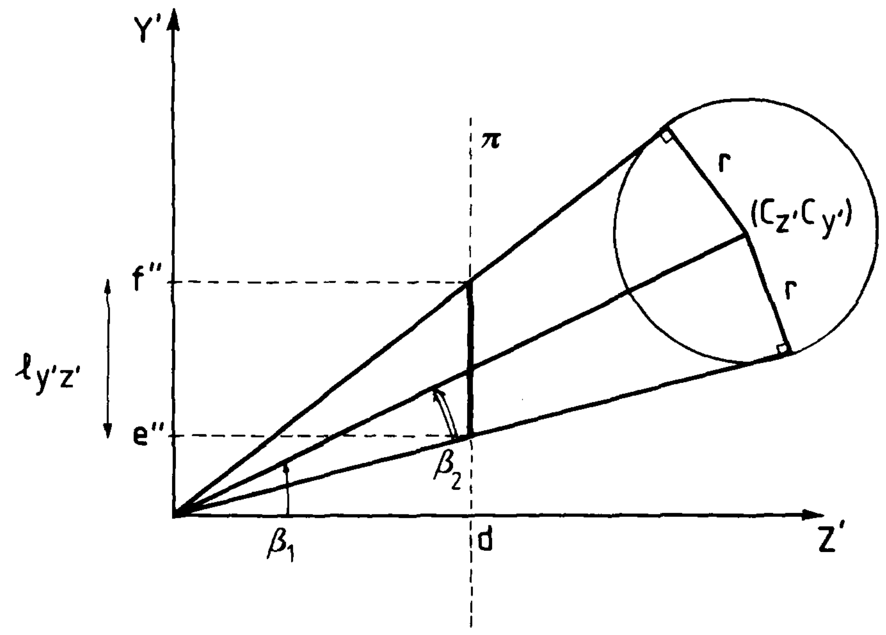
and

$$\alpha_1 = \tan^{-1}(c_{x'}/c_{z'})$$

$$\alpha_2 = \tan^{-1}[r/(c_{x'}^2 + c_{z'}^2 - r^2)^{\frac{1}{2}}].$$


Z'
r
$(C_{x'}C_{z'})$
r
α_1
d
π
α_2
O'
a''
b''
X'
$\ell_{x'z'}$

(a)


Y'
π
r
$(C_{z'}C_{y'})$
f''
r
$\ell_{y'z'}$
e''
β_2
β_1
d
Z'

(b)

FIG. 4.1

Similarly, with reference to Fig. 4.1(*b*), the length of the projection in the $Y'O'Z'$ plane is:

$$l_{y'z'} = f'' - e''$$

where

$$e'' = d \tan(\beta_1 - \beta_2)$$

$$f'' = d \tan(\beta_1 + \beta_2)$$

and

$$\beta_1 = \tan^{-1}(c_{y'}/c_{z'})$$

$$\beta_2 = \tan^{-1}[r/(c_{y'}^2 + c_{z'}^2 - r^2)^{\frac{1}{2}}]$$

The rectangular boundary in the Π plane is depicted in Fig. 4.2 where:

$$\gamma = d\, c_{x'}/c_{z'}$$

$$\delta = d\, c_{y'}/c_{z'}$$

and the following cluster clipping rule is readily deduced.

Cluster Clipping

A cluster will not be visible if any of the following statements are true:

(i) $\gamma > 0$ and $a'' > b$

(ii) $\gamma > 0$ and $b'' < -b$

(iii) $\delta > 0$ and $e'' > b$

(iv) $\delta < 0$ and $f'' < -b$

(v) $c_{z'} + r < d$.

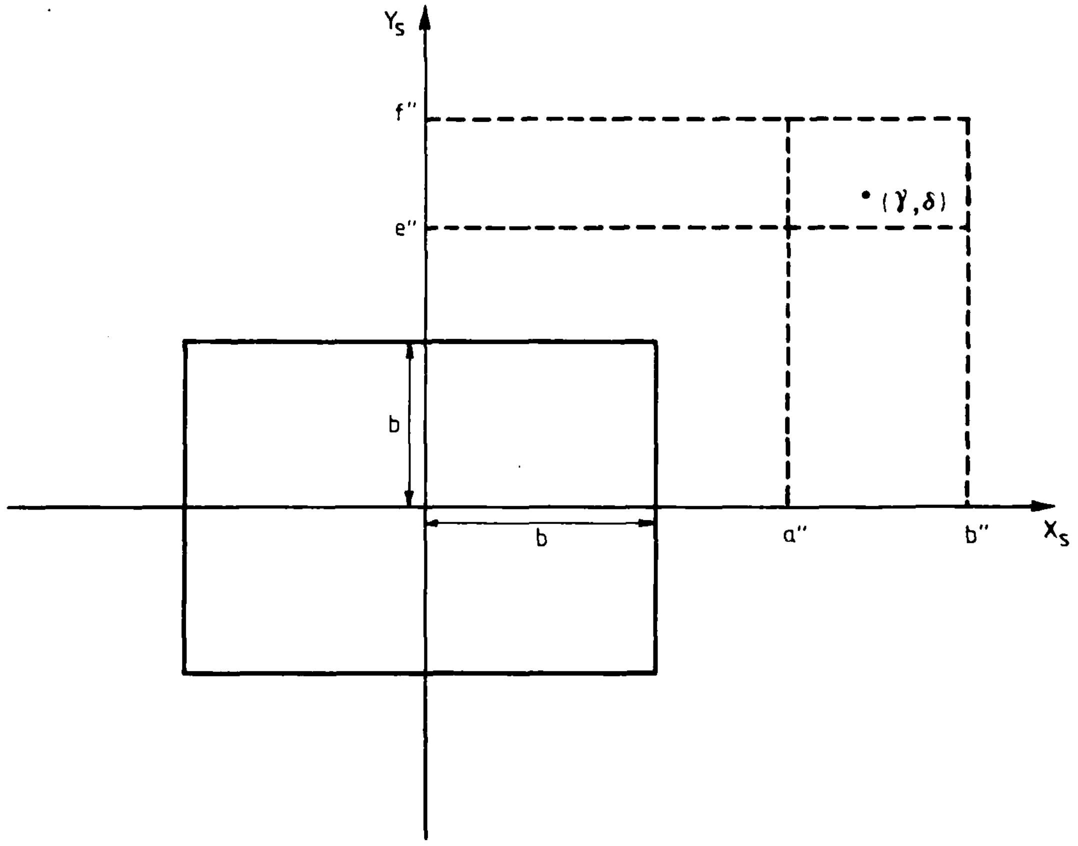


FIG. 4.2

In addition, cluster clipping can be extended to eliminate clusters whose projected image area is smaller than the resolution limit of the display device.

It is evident that if the environment is structured such that each cluster is made up of "smaller" clusters, then the recursive application of the cluster clipping algorithm will eliminate all the "out of view" objects from subsequent processing. In addition cluster clipping can be closely related to the concept of "level of detail"[2,19] selection which permits the selection, for each object in the environment, of a model of appropriate complexity consistant with the regional extent of the object, its relative position to the observer and the resolution of the display. Typically the ratio, enclosing sphere radius : perspective image radius, is a useful criterion for model selection, i.e. if the ratio is large, then a low detail model will be adequate whilst for small values a detailed model is necessary. The transition from one level of detail to the next must be such that the observer does not perceive the change from one model to another.

The level of detail concept is important in eliminating extraneous visual information and associated computational effort. The number of models associated with a given object will depend on the data base capacity and the object itself.

4.2 Edge Clipping

Edge clipping is the procedure that reduces a polygon surface extending beyond some three-dimensional viewing pyramid to a surface which does not extend beyond the pyramid boundary. The process clips off those parts of the polygon which lie outside the bounding volume. It is assumed, with little loss of generality, that the display screen is square; the associated viewing pyramid (henceforth referred to as "viewbox") is depicted in Fig. 4.3.

Consider now a transformed edge whose vertices lie outside the viewbox. Clearly, the edge can have two possible relationships with respect to the viewbox:

(i) it may lie completely outside the viewbox

or

(ii) some portion of it lies within the viewbox. In this case the edge must intersect two distinct viewbox boundaries.

In order to determine which of the above situations is applicable, it is necessary to determine the relative position of the edge vertices with respect to the viewbox. This can be readily accomplished by performing the following operations.[5-11]

$$\left.\begin{array}{ll} \textit{If:} & \textit{Then vertex } (x', y', z') \textit{ is:} \\ (h/d)z' - x' < 0 & \text{outside viewbox on right} \\ (h/d)z' + x' < 0 & \text{outside viewbox on left} \\ (h/d)z' - y' < 0 & \text{above viewbox} \\ (h/d)z' + y' < 0 & \text{below viewbox} \\ z' < d & \text{on wrong side of display screen} \end{array}\right\} \quad (4.2.1)$$

It is evident from the preceding remarks that if the vertices of an edge satisfy the same conditions then that edge is not visible and can be eliminated from further processing. On the other hand if the vertices satisfy two different conditions then it is possible that a portion of the edge is visible. To determine the relevant portion, the edge has to be truncated at the appropriate viewbox boundaries. The formulae to achieve this follow from elementary co-ordinate geometry and are stated below for an edge with vertices $r_1(x_1, y_1)$ and $r_2(x_2, y_2)$ in the $X_SO_SY_S$ plane.

However note that hither plane clipping (see Fig. 4.3) has to be performed in the $O'X'Y'Z'$ frame prior to perspective transformation since the final division destroys one bit of sign information. It is precisely this sign information that distinguishes points in front from those behind the display screen. Hence an edge with vertices $r_1(x_1', y_1', z_1')$, $r_2(x_2', y_2', z_2')$ will intersect the hither plane

in $(\hat{x}', \hat{y}', \hat{z}')$ where

$$\hat{x}' = x_1' + (x_2' - x_1')Q_H$$
$$\hat{y}' = y_1' + (y_2' - y_1')Q_H$$
$$\hat{z}' = z_1' + (z_2' - z_1')Q_H$$

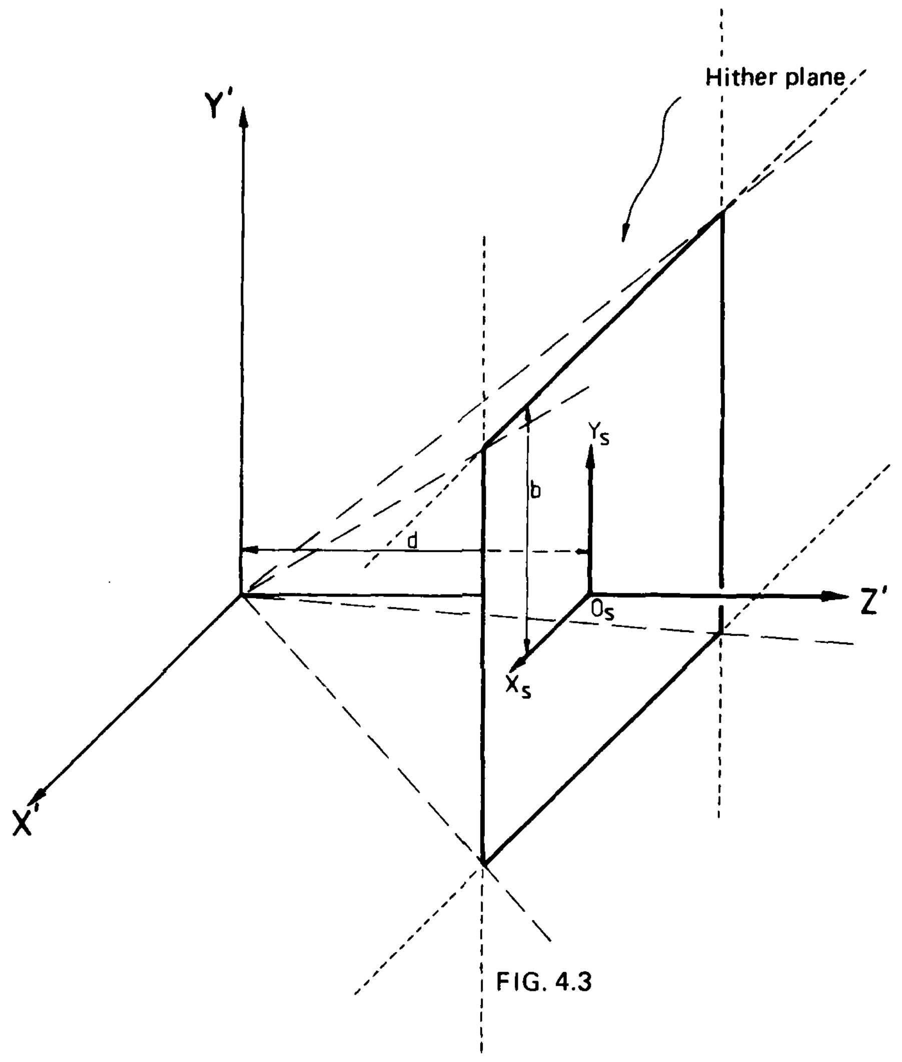


FIG. 4.3

and

$$Q_H = \frac{d - z_1'}{z_2' - z_1'}.$$

Referring to Fig. 4.4 the edge $v_1(x_1, y_1)$, $v_2(x_2, y_2)$ will intersect

(i) the right boundary line, $x_s = b$

in (x, y) where

$$x = x_1 + Q(x_2 - x_1)$$

$$y = y_1 + Q(y_2 - y_1)$$

where

$$Q = Q_R = (b - x_1)/(x_2 - x_1)$$

(ii) the left boundary line, $x_s = -b$

in (x, y) where

$$Q = Q_L = -(b + x_1)/(x_2 - x_1)$$

(iii) the top boundary line, $y_s = b$

in (x, y) where

$$Q = Q_T = (b - y_1)/(y_2 - y_1)$$

(iv) the bottom boundary line, $y_s = -b$

in (x, y) where

$$Q = Q_B = -(b + y_1)/(y_2 - y_1).$$

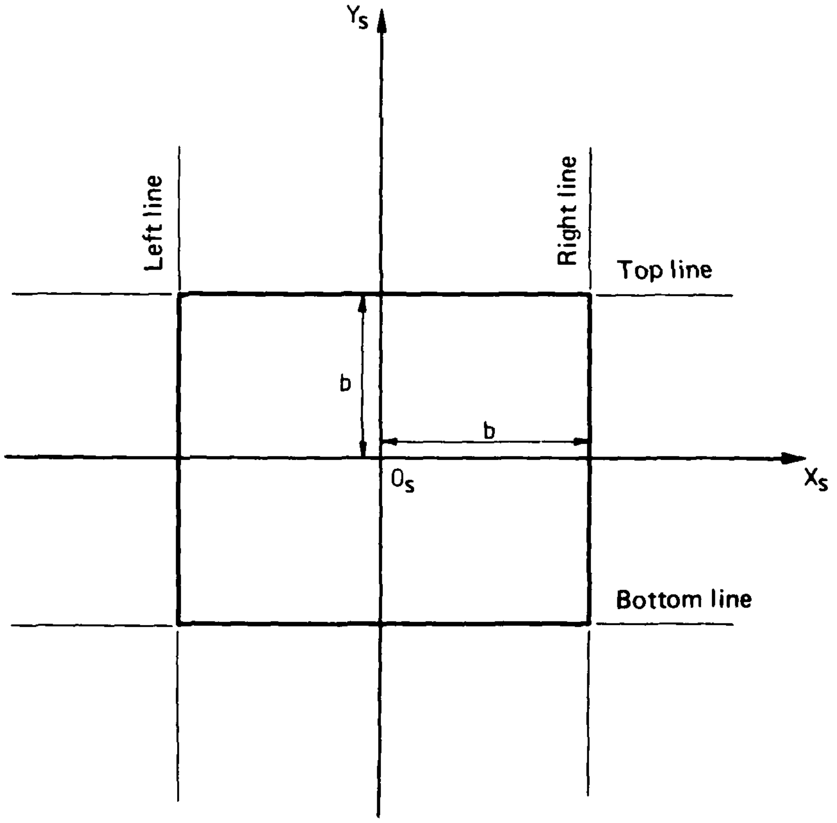


FIG. 4.4

Consider an edge, with vertices v_1, v_2, which is not wholly within the viewbox and let v_H, v_R, v_L, v_T, v_B denote the intersection of the edge with the respective boundary plane/lines as computed by the preceding formulae. Then the pair v_i, v_j, belonging to $\{v_1, v_2, v_H, v_R, v_L, v_T, v_B\}$, such that v_i, v_j (i) both lie on the segment $\overline{v_1 \; v_2}$ and (ii) are both within the viewbox, will define the clipped edge. Otherwise the edge in question is out of view. Repeated application of this process to the appropriate edges (as determined by (4.2.1)) will cull from the image those portions of the scene that extend beyond the field of view of the observer.

4.3 Polygon Clipping

The algorithm of Section 4.2 is applicable to polygon clipping by considering each edge of the polygon and clipping it against the viewbox. Unfortunately this approach involves complicated reasoning to determine where to add edges along the boundary. This is evident if one considers a polygon surrounding a viewbox vertex as depicted in Fig. 4.5. In this case two new edges have to be added which share a vertex at the corner. From the point of view of the edge clipping algorithm it is a tedious process to compute whether or not a polygon surrounds a corner of the viewbox.

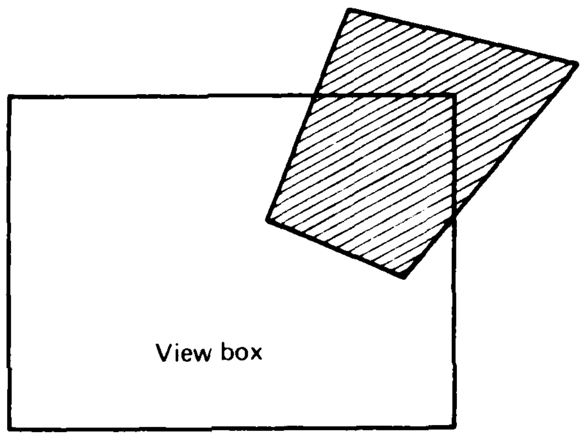
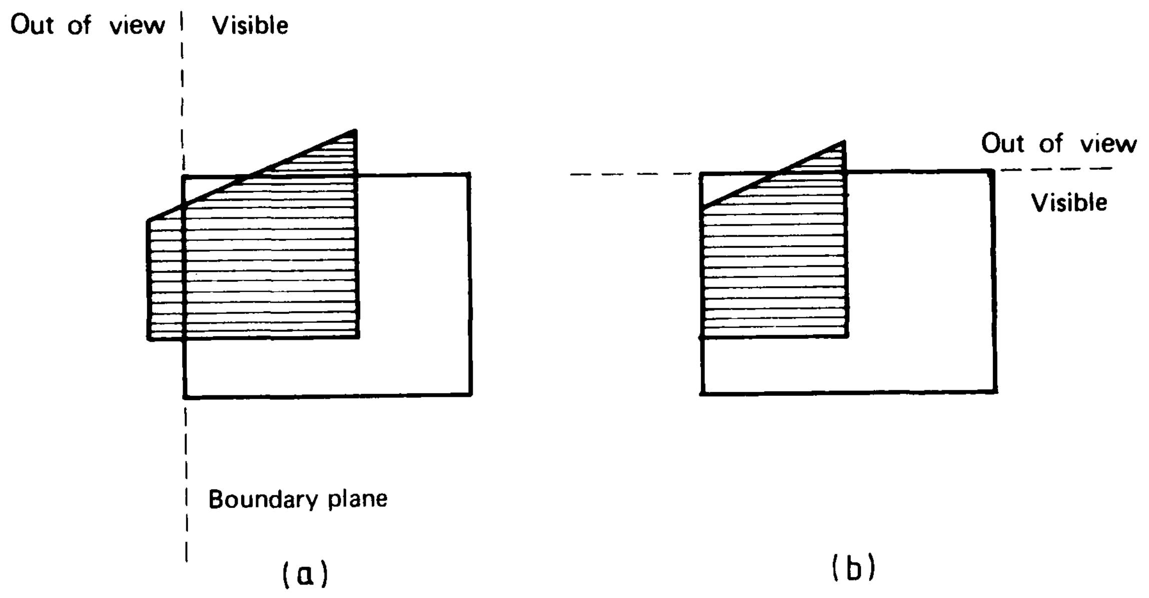


FIG. 4.5 POLYGON SURROUNDING VIEWBOX CORNER.

A clipping algorithm that circumvents this difficulty has been developed by Sutherland and Hodgman.[18] Basically the algorithm assumes that polygons are represented by an ordered sequence of vertices without repetition of first and last. This is in marked contrast to the preceding algorithm which regards polygons to be made up of edges. The clipped polygon has identical format, with new vertices introduced in sequence.

To illustrate the algorithm consider the clipping of a polygon against a boundary plane as depicted in Fig. 4.6(*a*). Clipping is accomplished by considering pairs of vertices (edges) and retaining only those vertices that are on the visible side of the boundary plane together with vertices created by intersection. The resulting polygon is shown in Fig. 4.6(*b*). Clipping the latter against the top boundary plane results in the final clipped polygon of Fig. 4.6(*c*). For a complete description of the algorithm the interested reader is referred to Reference 18.

In terms of computational effort polygon clipping is more expensive. The reason for this is twofold: firstly, each pair of vertices has to be tested against each boundary plane regardless of whether it specifies an edge which is entirely within the viewbox, and secondly, clipping a polygon against a boundary plane may increase the number of vertices describing the clipped polygon and hence increase the computation time for subsequent phases of clipping. These characteristics are due to the serial nature of the algorithm. This is in contrast to the edge clipping algorithm of Section 4.2 where the clipping of an edge against each boundary plane can be executed in parallel.



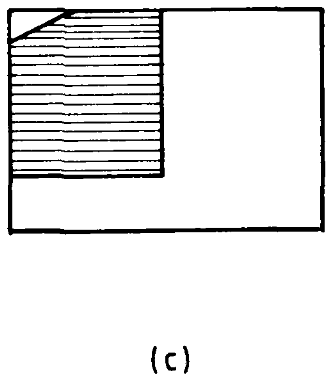


FIG. 4.6 POLYGON CLIPPING

5. HIDDEN SURFACE ALGORITHMS

Application of the preceding methods to three-dimensional objects will produce two-dimensional perspective images with one significant deficiency: namely that object faces, which are in reality occluded from the viewer by other objects, appear in the image. In practice the removal of these extraneous faces from the image is achieved by a "hidden surface algorithm". Typically such an algorithm has two underlying features: first the algorithm sorts through a collection of edges, faces or objects according to some criteria until the visible elements of the picture are determined, and secondly, utility is derived from the coherent nature of the scene being generated. (The term "coherent" will be used to describe the extent to which the scene being depicted is limited in spatial and temporal variability.)

In respect of the preceding remarks, the order and type of sort used, and the coherence properties exploited depends on the description of the constituent objects and the output medium for the generated picture. Since consideration here is restricted to objects modelled as convex polyhedra and raster scan display devices, the discussion in this section is limited to hidden surface algorithms that fall into the category commonly termed "scan line algorithms".

5.1 Back Face Elimination

A face of an object, hidden by its own volume from the observer is termed a "back face". Although most of the scan line algorithms discussed in the next section process and eliminate back faces in exactly the same way as they do surfaces occluded by other objects, it may prove more efficient to remove back faces prior to polygon (or edge) clipping.

To determine whether an object face is or is not a back face with respect to the observer, it is necessary to compute:

$$\Delta = \frac{c' \cdot n'}{c' \quad n'}$$

where c and $\underline{n}$ are the face centroid and outward normal vector respectively

and it can be concluded[27] that if

$$\Delta \geq 0$$

then the face is occluded from view by its own volume. Since c and $\underline{n}$ can both be determined from vertex data, the preceding computation can be readily included in any image synthesis procedure.

5.2 Review of Some Hidden Surface Algorithms

The purpose of this section is to review some existing hidden surface algorithms which have real time capability, and to determine the dominant requirements for an efficient algorithm. For a full description of ten hidden surface algorithms the interested reader is referred to the excellent article by Sutherland *et al.*[6]

In a broad sense hidden surface algorithms can be classified as:

(*a*) *Object Space Algorithms;* these utilize the observer frame ($O'X'Y'Z'$) description of the scene to compute the visible surfaces. The computation is performed with sufficient precision so that the synthesized image is correct even if enlarged many times.

(*b*) *Image Space Algorithms;* these perform the computation in the image space $O_SX_SY_S$ and require only sufficient accuracy to match the resolution of the display device.

(*c*) *List Priority Algorithms;* these work partly in the observer frame and partly in the image space.

In terms of performance, the execution time for an object space algorithm increases as a function of the scene complexity, whilst for image space methods it has an upper bound which is independent of scene complexity. The reason for the latter property is that the resolution limit of the display device provides a convenient culling criterion for object images, i.e. images smaller than the resolution limit can be eliminated from further processing by the hidden surface algorithm. It is for this reason that algorithms based on image space computations (i.e. image space and priority list algorithms) are the only methods under consideration for real time picture generation. Hence attention will be restricted to this class of algorithms.

Image space algorithms can be conveniently classified according to the order in which the image space polygons are sorted. The basic difference is whether the depth sort is performed before the horizontal and vertical sorts or vice versa. There is one exception to this scheme and that is the area based algorithm of Warnock.

I. Warnock Algorithm[21]

The general idea behind Warnock's algorithm is to sub-divide the image space picture into portions (typically squares) termed "sample windows". Warnock declares a sample window to be homogeneous if: (i) no faces fall within the sample window or (ii) one face completely covers the window and is nearer the viewpoint than every other face that falls within the window and hence no further processing is necessary. If a sample window is not homogeneous then it is divided into four smaller sample windows and each of these is examined analogously. The sub-division process terminates at the resolution limit of the display device.

The Warnock algorithm has two major defects:[6] first, the relationship between object faces and display screen area is difficult to compute, and second, no suitable output device exists that can output information an area at a time. In fact, the apparent random way the algorithm partitions the picture requires that the image be displayed on a random-scan display device.

II. Depth Sort First

The most significant algorithms belonging to this class are due to Schumacker *et al.* and Newell *et al.* Both algorithms compute, in object space, a priority list of polygons which is subsequently utilized in image space to eliminate hidden surfaces.

(a) Schumacker's Algorithm[7]

The priority list of potentially visible polygons is establised with the aid of a polygon clustering concept, the main features of which are:

(i) for linearly separable clusters the cluster priority depends on the location of the viewpoint relative to the separating planes;

(ii) the face priority within a cluster depends only on the topology of the cluster, is independent of the viewpoint and hence can be computed *a priori*.

The calculation of the priority of several clusters is demonstrated by Fig. 5.1(*a*).[6] If the viewpoint lies in region *C*, then cluster 3 will have priority over clusters 1 and 2 whilst cluster 2 will have priority over cluster 1 by virtue of the fact that for a given viewing position in region *C*, cluster 2 is closer to this viewpoint than cluster 1. Extending this process to the other three regions a tree structure (Fig. 5.1(*b*)) can be constructed which shows how the relationship of the viewpoint to the two separating planes produces one of four possible orderings of the three clusters. The concept extends to an arbitrary number of linearly separable clusters.

The computation of face priority (with reference to Fig. 5.2(*a*)) requires computing whether face *A* can, from any viewpoint, hide face *B*. If this is so then face *A* has priority over face *B*. Hence face priority within a cluster will be independent of viewpoint since different back faces are eliminated for each viewpoint leaving the remaining faces ordered correctly (see Fig. 5.2).

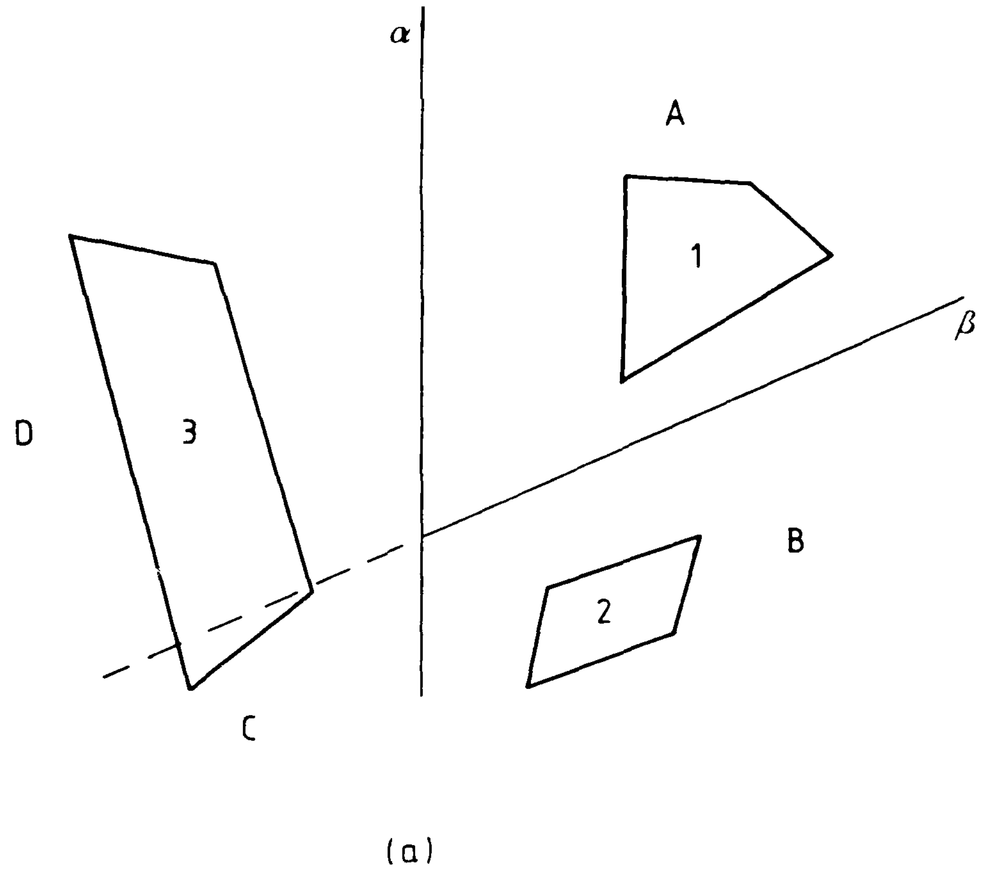
The generation of the processed image is achieved in two steps. Firstly, for the given viewpoint the back faces of objects are eliminated and a priority list of polygons is computed by comparing the viewpoint to the separating planes. Then for each scan line the edges intersected are determined and utilizing the priority list of polygons the visible surface at each pixel is determined.

Although the computation of the cluster priority list is potentially a costly process (since it may involve the search of an extensive tree structure) the algorithm is able to capitalize on frame coherence; i.e. the cluster priority list remains unchanged from one frame to the next unless the viewpoint crosses one of the separating planes.

The notion of separating planes and list priority have been utilized in Reference 2 (p. 170).

(b) Newell's Algorithm[6]

The principal significance of Newell's algorithm is the development of the concept of "over writing".



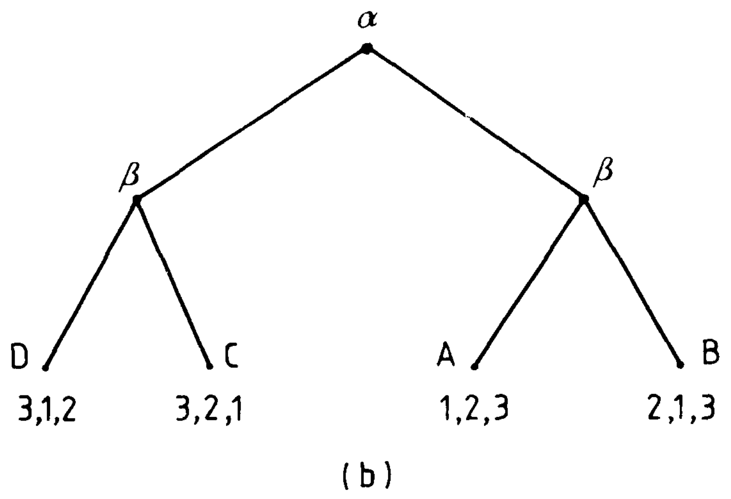


FIG. 5.1

(a) α, β are the separating planes
(b) The modes represent the separating planes. The branches represent the relevant regions with respect to the planes.

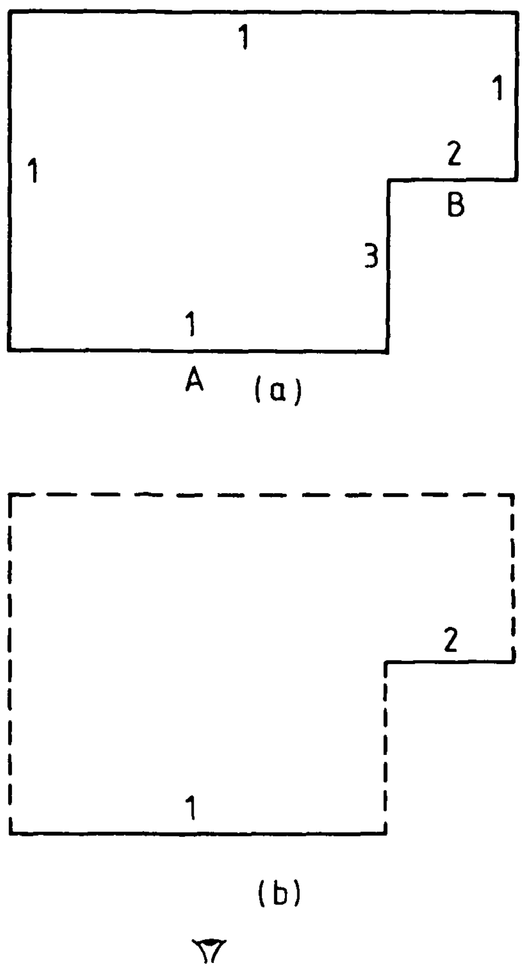


FIG. 5.2

The algorithm first sorts the collection of faces into a priority order according to their distances from the viewing plane. This information is then utilized to establish the visibility priority of polygon segments on a given scan line. Those segments with a lesser priority are written into a scan line buffer before writing those with a higher priority. Once the entire priority list has been processed the buffer will have a correct hidden surface view since higher priority faces have overwritten lower priority ones. Whilst there is considerable overhead in writing into buffer segments that may eventually be obscured, some high quality pictures have been produced by the algorithm.[8]

A technique similar to Newell's, but based on a barrel memory of MOS shift registers, has been proposed by Sale and Bromley.[9] Although the predicted efficiency seems promising from the point of view of real time image generation, the technique is yet to be implemented.

III. Depth-Last Sort

The algorithms using this sorting order have been devised by Watkins,[3] Bouknight[1] and Wylie *et al.*[10] and have a close resemblance to each other. All three procedures first sort the visible edges in order of their occurrence in the Y_S direction (this is termed the "Y-sort"). Then for each scan line the edge intercepts are sorted in the X_S direction (X-sort) and finally a Z-depth search determines the visible face at each pixel. The depth search is performed last under the premise that the first two sorts will decrease the number of depth comparisons needed to establish the visible surface.

Each of the algorithms utilizes scan line coherence. That is, edges that intersect a scan line are likely to intersect the next one. Hence, as processing for each scan line begins, the Y-sorted list is examined to find new edges that begin on the scan line and these are then added to the ones already entered. Edges that terminate on this scan line are discarded.

The algorithms next examine the list of "active" edges to compute which faces are visible in each portion of the scan line. This is accomplished by partitioning the scan line into smaller segments called "sample spans", the extent of a span depends on the particular algorithm being used. A depth comparison amongst all the faces whose edges cross the span determines the visible content of the span.

The depth comparison computation can be made more efficient by the observation made in Ref. 10 concerning *depth coherence*: if the same faces are intersected by one scan line as by the previous one, and if the X-crossing order of the edges are the same, then the depth comparison computation need not be repeated. The same faces will be visible as before although their extent in X may be different.

5.3 Sorting

Sorting is central to the hidden surface problem and hence considerable care must be exercised in selecting sorting methods which will conserve computation time by capitalizing on the coherence available in the scenes being rendered. Three sorting algorithms, which have been incorporated in a number of hidden surface algorithms, will be reviewed. For this purpose it will be assumed that records $R_1, \ldots, R_N$ are to be sorted in non-decreasing order of their keys, $K_1, \ldots, K_N$.

Quicksort: This is a comparison exchange scheme ideally suited for serial computation. The execution of the algorithm is as follows:[12] keep two pointers i and j with $i = 1$ and $j = N$ initially. Compare $K_i : K_j$ and if no exchange is required decrease j by 1 and repeat the process. After an exchange first occurs, increase i by 1 and continue comparing $K_i : K_j$ and increasing i until another exchange occurs. Then decreasing j again, continue the process until $i = j$. As an illustration of the procedure consider the following example

Given:	***503***	512	908	897	275	426	154	509	***677***
									Decrease j
1st Exchange:	***154***	512	908	897	275	426	***503***	509	677
									Increase i
2nd Exchange:	154	***503***	908	897	275	426	***512***	509	677
									Decrease j
3rd Exchange:	154	***426***	908	897	275	***503***	512	509	677
									Increase i
4th Exchange:	154	426	***503***	897	275	***908***	512	509	677
									Decrease j
5th Exchange:	154	426	***275***	897	***503***	908	512	509	677
									Increase i
6th Exchange:	154	426	275	***503***	***897***	908	512	509	677

(K_i and K_j are shown in heavy italic type.)

Observe that by the time $i = j$ the original record R_1 will have moved into its correct place and that there will be no greater keys to its left, nor smaller keys to its right. Hence the resulting list can be partitioned into two simpler sorting problems. Furthermore each comparison will involve the original key K_1, which may be kept in a register and need not be stored until the end of the computation. The computation time[12] associated with a Quicksort is of the order of $N \log_2 N$ for a random list and of the order of N^2 for a list nearly in sort. Unlike most sorting methods a Quicksort is more efficient for a disordered list.

Radix sort: This is a procedure based on the digits of the keys and requires 2 N memory locations for processing N records. The sort is carried out as follows:[12] start with a distribution sort based on the least significant digit of the keys (in radix M) and partition the last N memory locations (termed auxiliary store) into M segments: each segment corresponding to one of the M digits and the number of memory locations in each segment being determined by the distribution sort. Each record is then placed, according to its least significant digit, in the appropriate segment in the auxiliary store. Carrying out another distribution sort on the next least significant digit, move the records back into the original input area, repeating the procedure until the most significant digit.

As an illustration consider the preceding example:

Input area:	503	512	908	897	275	426	154	509	677	703
Counts for units digit distribution:	0	0	1	2	1	1	1	2	1	1
Storage allocation based on distribution:	0	0	1	3	4	5	6	8	9	10
Auxiliary area:	512	503	703	154	275	426	897	677	708	509
Distribution for tens digit:	4	1	1	0	0	1	0	2	0	1
Storage allocation:	4	5	6	6	6	7	7	9	9	10
Input area:	503	703	908	509	512	426	154	275	677	897
Distribution for hundreds digit:	0	1	1	0	1	3	1	1	1	1
Storage allocation:	0	1	2	2	3	6	7	8	9	10
Auxiliary area:	154	275	426	503	509	512	677	703	897	908

Observe that the number of passes required to sort a list depends on the radix and the number digits in the keys but is independent of N. Moreover it can be shown that for random data the sorting time is of the order of N.

In terms of the implementation (memory requirement, hardware) both the Quicksort and the radix sort are most efficient for large N and random data. In the context of scan line algorithms, these techniques have been utilized in performing the Y sort for which they are ideally suited. The reason for this is that the finite display resolution coupled with edge clipping implies that a radix sort need only consider nine or ten bits of a 16-bit key field. This is not the case for the Z sort since the distribution of depths in any image will, in general, be unknown. This fact is part of the reason for the attraction of depth-sort last scan line algorithm.

Bubble sort: This is perhaps the most obvious way to sort by comparison and exchange. The process is as follows: compare K_1 and K_2, interchange R_1, R_2 if keys are out of order; repeat for R_2 and R_3, R_3 and R_4, etc. During the first pass of this process the record with the largest key will move to become R_N. Repeated passes will move appropriate records into positions R_{N-1}, R_{N-2}, etc., until the entire list is in sort. The time required to sort random data is of the order of N^2 making it unsuitable for large N. Nevertheless it is well suited to performing the X sort from one scan line to the next.[1] The reason being that line to line coherence implies that edge order changes infrequently and when it does it normally only involves adjacent pairs whose order need only be interchanged to restore the correct ordering. Thus not only is the "bubbling" operation required infrequently, but the "bubble" does not have to move very far in the list.

6. IMAGE QUALITY

In scope, image quality encompasses those attributes of detail without which significant differentiation could be made between the physical and synthesized scene irrespective of image content. In this sense the parameters of interest for CGI flight simulation visuals are shading, fading, shadows and quantization effects.

6.1 Shading Functions

The essence of shading computer synthesized images is the computation, via a shading function, of the intensity and colour of each visible point of an object from the characteristics of the light source, the nature of the object and the position of the observer. In addition, proper shading depends on a further number of objective factors, of which the following are significant but by no means exhaustive:

(*a*) diffuse and specular reflections;
(*b*) multiple light sources including inter-surface reflections;
(*c*) haziness (or blurring) as a function of distance;
(*d*) shadows.

A number of shading functions have been proposed that accommodate some of the above features. Although only one of them refers specifically to flight simulation visuals it is instructive to review these approaches.

Romney[22] assumes that a point light source is located at the observer's eye and employs the empirically derived shading rule:

$$\frac{\cos^2 \theta}{r^4} \qquad (6.1.1)$$

where θ is the angle of view measured from the surface normal;
r is the distance between the observer and the source

to render shaded images of diffuse surfaced objects for a monochrome raster display. The resulting pictures have certain desirable features, namely that for r fixed, surfaces pointing away from the observer appear less bright than those facing the observer. In contrast, for θ fixed, the term $1/r^4$ drops off quite rapidly for small r and changes move slowly for large r. The effect is to render closer objects bright and the more distant objects with a less bright but nearly identical shade. Although the $1/r^4$ term leads to a "miner's lamp" effect, (6.1.1) could be useful as a first approximation to a more realistic shading function, in the generation of night scenes comprised of diffuse surfaces illuminated by directional point light sources.

In the case of day scenes illuminated by the sun and the sky (6.1.1) is not applicable. Instead each object surface is characterized by a predetermined shade about which the assigned shade varies according to the relative orientation of the surface with respect to the sun.

A relatively simple implementation of this has been reported in Reference 2. It involves computing the angular position of the sun with respect to each surface normal and the cosine of the angle is used to modify 60% of the pre-assigned shade whilst the remaining 40% is assumed due to diffuse illumination from the sky and reflections from other surfaces.

For the inclusion of specular reflections under point source illumination, Warnock[21] and Phong[13] have suggested the addition of a term of the form:

$$\frac{W(\phi)[\cos(\theta - \phi)]^m}{d^2} \qquad (6.1.2)$$

(where θ is angle between the line of sight and the surface normal (in the sense of Fig. 6.1), ϕ is the angle of incidence;
$W(\cdot)$ is a function of the ratio of reflected and incident light at angle of incidence ϕ;
d is the distance between the observer and the object surface).

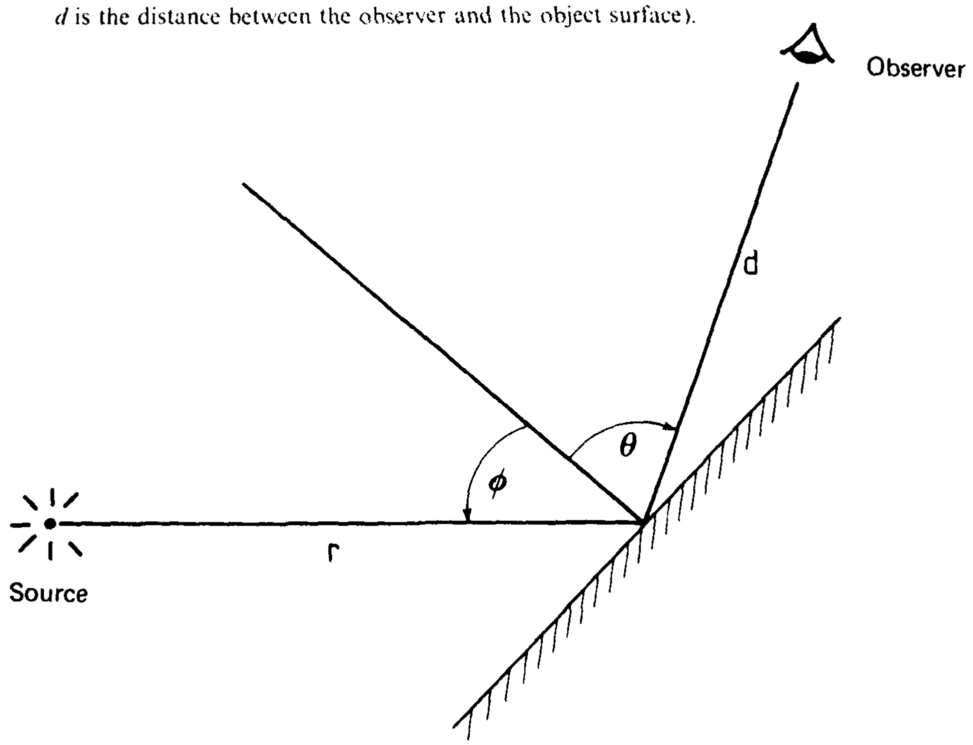


FIG. 6.1

to a shading function such as (6.1.1). The function $W(\cdot)$ and m are chosen empirically, although in general a large value of m is required to characterize a glossy surface. This follows from the fact that the greatest amount of specular reflection occurs in the region $\theta = \phi$ and hence the inclusion of $[\cos(\theta - \phi)]^m$ which falls off rapidly outside the region.

6.2 Horizon Fading

Although the approach to shading embodied in (6.1.1) and (6.1.2) takes into account the nature of the surfaces, the position of the source and the observer, no allowance is made for the observed atmospheric effect of the fading of colours to a uniform fog-grey at the horizon. The result is that the generated image has a "cartoon-like" quality which in terms of flight simulation reduces depth perception.[14]

As reported in Reference 2 the remedy for this deficiency is to desaturate the colours of all objects in the scene as an exponential function of distance to the observer. An example[14] of such a computation is:

$$C_f = FC + (1 - F)G$$

where C_f = the resultant shade;
F = range factor, e^{-kd}, in which k = attenuation coefficient, which may depend on the altitude and elevation angle (with respect to observer), and d = range of object from observer;
C = assigned shade of object (as determined by the appropriate shading function);
G = fog/haze colour.

6.3 Shading of Curved Surfaces

It has been implicit in the preceding discussion that the objects had planar polygonal faces. For this class of objects it has been demonstrated[13] that a shading rule which embodies terms of the form of (6.1.1) and (6.1.2) together with a fading function will produce realistic pictures. The centroid, normal and intrinsic colour (spectral reflectance) of each face constitute the required information for the computations.

The application of the above approach to curved surfaces approximated by planar polygons will result in a shade discontinuity at polygon boundaries destroying the smoothness of the surface. One might expect that the problem would be alleviated by reducing the size of the approximating polygons. Unfortunately, because of visual perception effects, the reduction of polygon size is not as beneficial as might be expected.[13] The principal phenomenon responsible for this is the Mach Band Effect which states that intensity distortion results from discontinuity in the first derivative of the shading function.

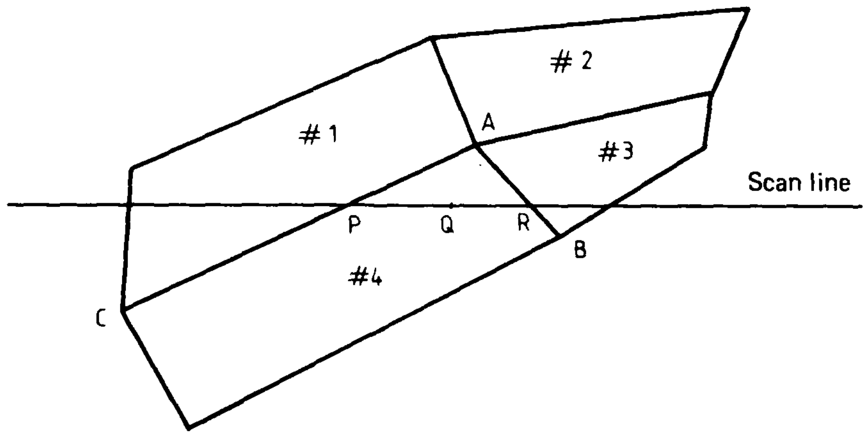
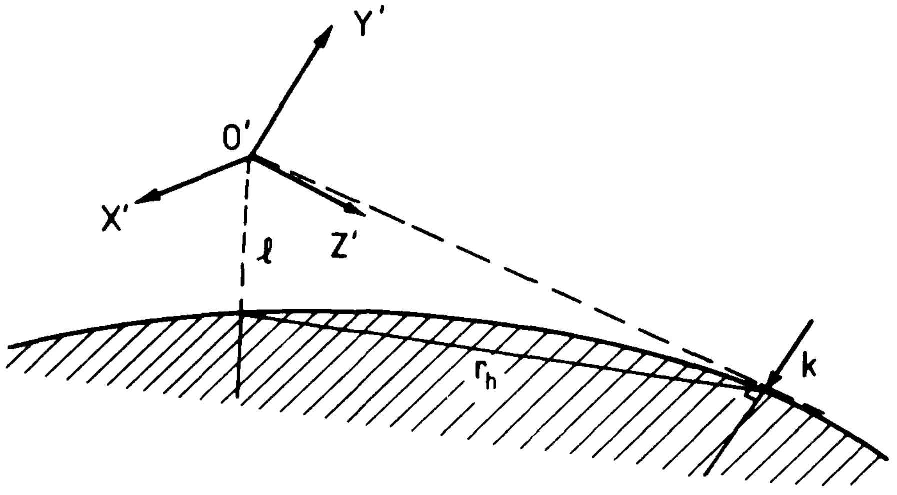
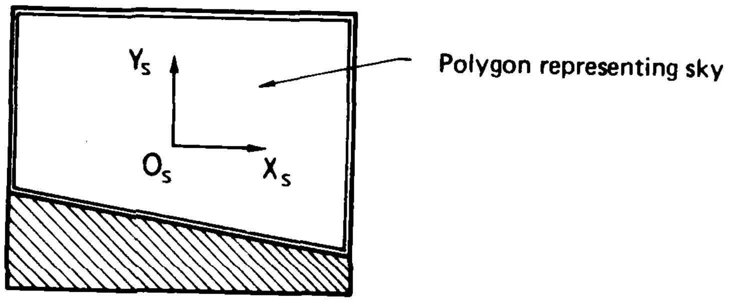


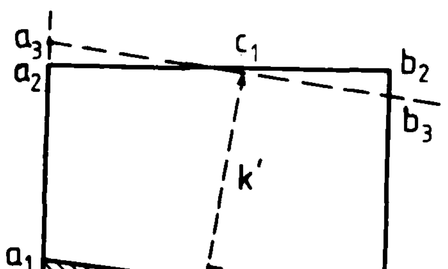
FIG. 6.2



(a)



(b)



k' is the image of k in (a).

(c)

FIG. 6.3

To restore the smooth appearance of curved surfaces approximated by polygons, Gouraud[15] observed that the value of the shading function as well as its derivative has to be continuous across polygon boundaries. Gouraud developed a simple technique, based on shading values at polygon vertices and interpolation, to restore smoothness. To illustrate the method consider the computation of the shade at point Q in Fig. 6.2. Firstly the shade at vertices A, B, C is computed, with the normal at each vertex being taken as the average of the normals of the adjoining faces. Then linear interpolation between the shade at A and B, and A and C will give the shade at points P and R respectively. A second interpolation between P and R gives the desired value at Q.

The Gouraud shading technique has been implemented in real time[2] and static[17] generation of a variety of images of great realism. In the context of scan line algorithms, the technique will require that each vertex be linked to its surrounding faces in order to compute the vertex normals. In general this additional computation will deteriorate the real time performance of a hidden surface algorithm with the result that the number of curved objects (e.g. aircraft fuselages) present in a scene will be limited.

A useful modification of the preceding approach is applicable to the shading of long objects (such as runways) for which the face centroid is an inadequate measure of depth. Specifically the shade calculation at each vertex is based on the surface normal and the vertex depth such that the shade at any interior point is again determined by two linear interpolations.

6.4 Representation of the Sky

For a scene observed near the surface of the earth the horizon can be represented as a circle of radius

$$r_h = \sqrt{2lr_e}; \; l/r_e \text{ small}$$

where r_e is the radius of the earth and l is the altitude of the observer, as illustrated in Fig. 6.3(a). The image of the visible portion of the horizon is a straight line which forms one of the edges of the polygon representing the sky (Fig. 6.3(b)). In the scan line algorithm sense this polygon is assigned the lowest priority and hence occludes no other object from the observer.

A realistic representation of the sky requires that the colour of the sky desaturate in the vicinity of the horizon. (The region is specified by k in Fig. 6.3(a) and its value has to be determined empirically.) One way of rendering this desaturation is to use Gouraud's shading technique (Section 6.3). To illustrate consider the situation depicted in Fig. 6.3(c) and assume the colour of the sky is B whilst the desired grey shade at the horizon is G. Assign the colour B to vertices a_3, b_2, b_3, c_1 and colour G to vertices a_1, b_1, whilst the colour at a_2 is determined by interpolation between a_1 and a_3. The polygon $(a_1, a_2, c_1, b_3, b_1)$ is shaded by Gouraud's method whilst the polygon (b_2, b_3, c_1) is coloured uniformly B. Other images involving the sky can be handled analogously.

6.5 Shadows

There are two types of shadows that result from point source illumination of a scene:

(i) A *projected shadow area* is a visible surface or portion of a visible surface from every point of which a vector to the point light source intersects a solid object (area A, Fig. 6.4);

(ii) A *self-shadow face* is a visible scene face which is orientated away from the point light source (face B, Fig. 6.4).

In a day scene the presence of a diffuse source (the sky) results in object edges being visible within shadows. Moreover the perceived colour of an area in shadow is increased in saturation but reduced in brightness.

Although shadows are desirable from the point of view of realism, perception, and simulation of time of day, they have received minimal attention. This is due in most part to the additional computational effort required to determine shadow areas. The computational problem is accentuated by the fact that a shadow of one object may interact with another object and its shadow as illustrated in Fig. 6.5. Hence in addition to depth relationships amongst scene objects one needs to have a knowledge of the relative orientation of objects amongst themselves and with respect to the light source for shadow generation.

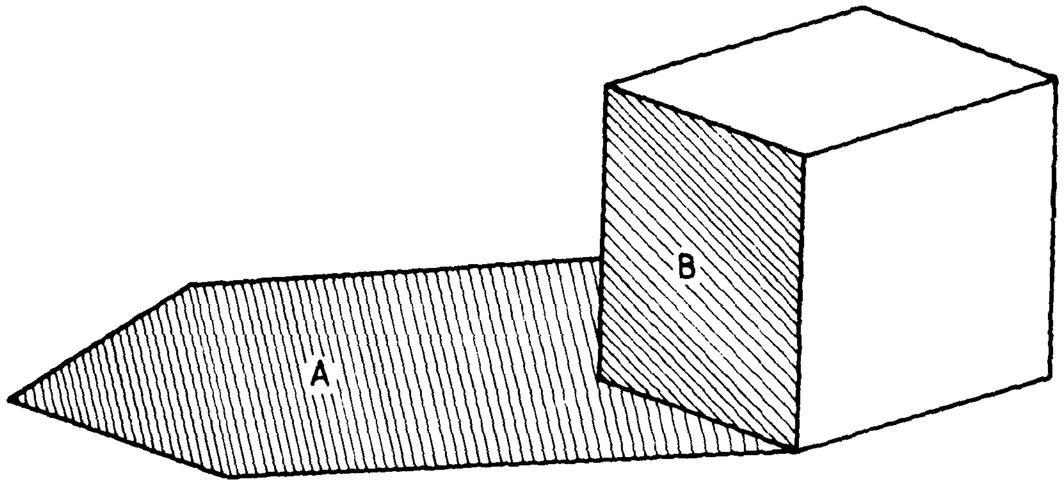


FIG. 6.4

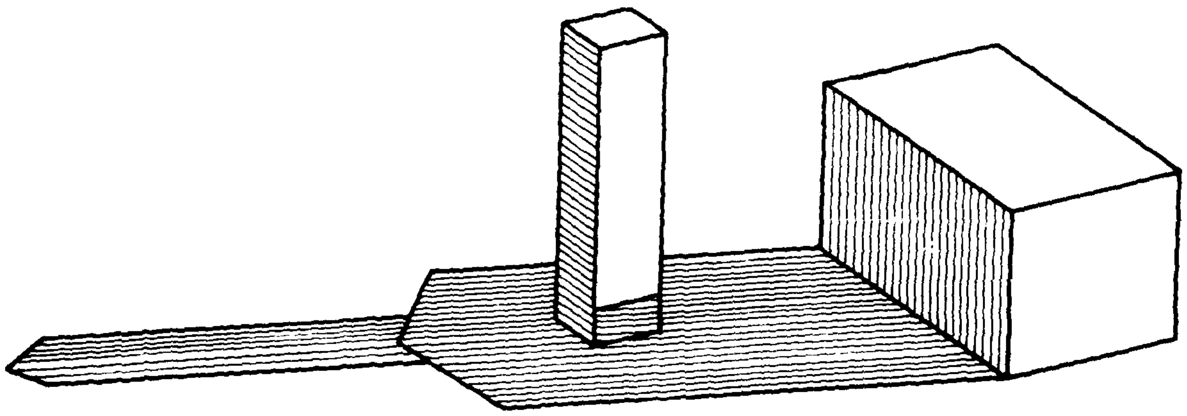


FIG. 6.5

6.6 Quantization Effects

These effects arise from the discrete nature of a raster scan display, and in the case of dynamic pictures, are responsible for distraction, loss of realism and false motional cues. The particular quantization effect present in a picture will depend on the particular rule employed to decide whether a pixel is within a surface or not. Typically if the centre of a pixel falls within a surface then the whole pixel is assigned the shade of that surface.

The resulting quantization effects can be categorized as follows:

(i) *Step Effect.*[2] Edges that are straight appear discontinuous. This is particularly apparent on near vertical and near horizontal edges separating faces of significantly different shade (Fig. 6.6(*a*)).

(ii) *Face Crawling.* Consider a face $2\frac{1}{2}$ elements wide with vertical edges. As the face moves across the screen the face width will alternate between two and three elements causing the face to appear to move by advancing to the right then receding from the left.

(iii) *Face break-up.* Consider a near horizontal face one scan line in width. As it intersects a given scan line, there will be groups of elements whose centres fall within the face—these will receive full colour. However on the next scan line the elements with centres within the face will be laterally displaced causing the face to appear to be made up of discrete segments (Fig. 6.6(*b*)).

(iv) *Blinking.* A face of less than one element width in its smallest dimension may be so orientated that it will periodically contain no element centre, in which case it will disappear.

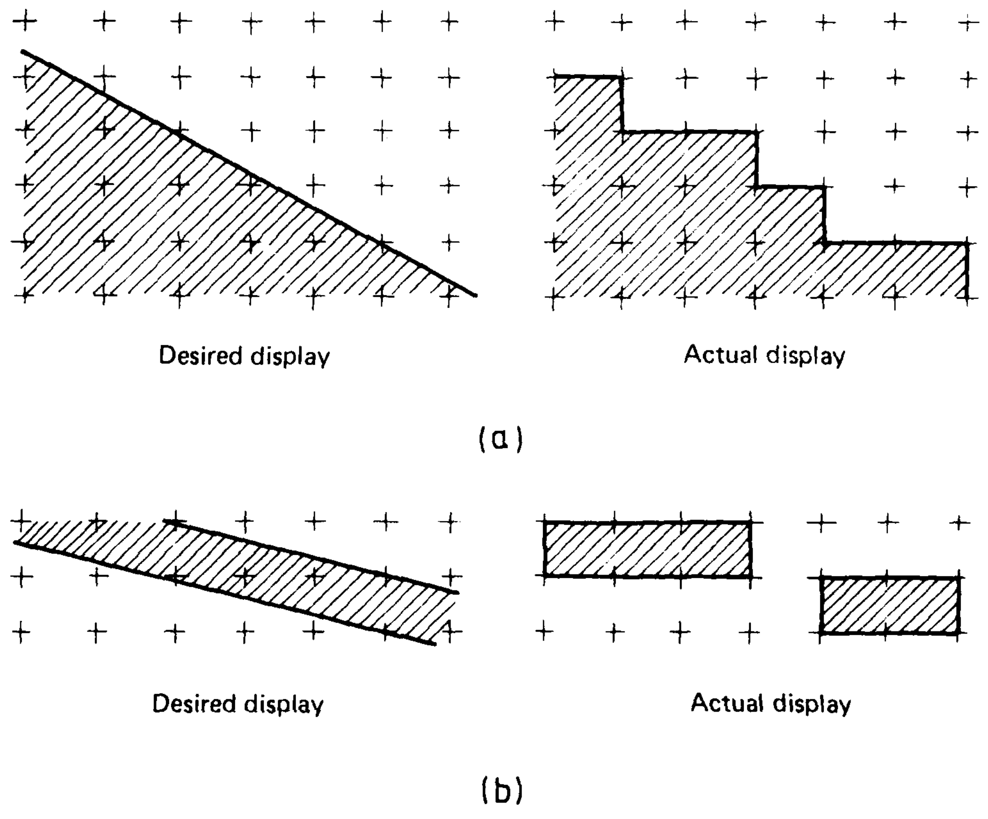


FIG. 6.6 QUANTIZATION EFFECTS

An effective solution to these problems[2] is to display each pixel which is cut by an edge as a blend of the colours within it, the percentage of each is determined in accordance with the area involved. One way to mechanize[23] this is to sub-divide each scan line into *k* sub-lines and each element into k sub-elements. Then the assigned colour of a pixel is the average of the k^2 sub-pixels. If more than two edges cross a given pixel then only the two edges associated with the two closest surfaces need to be considered in the smoothing operation.

That the above procedures provide an adequate remedy to the effects of quantization are evidenced by the synthesized images reported in Refs 2 and 23. A comprehensive review of the "state of the art" in the elimination of quantization effects can be found in Reference 24.

6.7 Night Scenes

The "world" of a night scene simulation typically consists of lights,[20] gross terrain and cultural features with low detail and low contrast, and rather concentrated areas at airports where faithful rendition of detail is important. Commercial CGI systems dedicated to night visuals have employed high resolution beam penetration CRT displays, resulting in high quality images, whilst until recently, raster scan based night scenes have been only moderately successful due to resolution limitations.[20] That is, if a single pixel represents a point light source, the motion of the source will result in objectionable discrete motion on the display which is accentuated by the high contrast between background and lights.

The general approach, which is akin to that of the previous section, to overcome this problem is to spread each light source over adjacent pixels smoothing the motion of light sources. Nevertheless the number of pixels over which a light source is spread has to be chosen carefully since the effective size of the light source image will increase.

7. CONCLUDING REMARKS

The topics discussed in the preceding section represent a broad spectrum of techniques for real time computer synthesis of images. However if capability and effectiveness is to be enhanced then a number of areas and possible alternatives need to be investigated. These delineate into environment modelling, hidden surface algorithms and image quality.

Environment Modelling. The modelling of objects by polyhedra is a convenient way of numerically characterizing the visual environment. Economy insists that the number of polygons be minimized, while quality of representation insists that the approximation render the image faithfully. For efficient environment modelling, quantitative guidelines, accommodating these two criteria, have to be found for choosing the approximating polygons. Moreover, it may prove convenient in certain situations to model curved surfaces by quadratic or higher degree surface patches. As evidenced by published results (Refs 16 and 25) pictures of remarkable quality can be produced by the method although the real time computational load is at present a limiting factor.

Hidden Surface Algorithms. The outstanding feature of the algorithms discussed is the use of coherency as a basis for efficiently computing the image. For example Schumacker makes use of cluster coherence to reduce the per frame computing cost (at the expense of additional environment preparation). On the other hand the scan line algorithms make use of lateral coherency to reduce the number of surfaces under consideration at any one point on the display. The latter is useful in that the number of depth comparisons necessary are small which is in contrast to the object space algorithms which depth sort the image polygons first.

The merging of ideas from object and image space methods may yield more efficient versions of a number of the algorithms discussed in Section 5.2. For example:

(i) Newell's algorithm might make use of frame to frame coherence by saving the priority order from one frame to the next;

(ii) Bouknight's and Watkins' algorithms might make use of Schumacker's clustering concept to reduce the final depth comparison computation.

Image Quality. Whilst shading and edge smoothing are the most significant aspects of picture quality there are other areas, relevant to image recognition and cues,[15] that warrant further investigation. In particular, for low level attack and nap of the earth flight simulation, texture, reflections and shadows, are requisites for the accurate and detailed rendition of man-made and natural features, including buildings, contoured terrain and gradation of surfaces.[26] Part of the problem with synthesizing these features is that they are difficult, as in the case of texture, to characterize numerically.

REFERENCES

1. Bouknight, W. J., A Procedure for Generation of Three-Dimensional Half-Toned Computer Graphics Presentations. Comm. ACM, Vol. 13, No. 9, Sept. 1970, pp. 527-536.

2. Beardsley, H., *et al.*, Advanced Simulation in Undergraduate Pilot Training. AFHRL-TR-75-59(V). U.S. Air Force Human Resources Laboratory, Nov. 1975.

3. Watkins, G. S., A Real-Time Visible Surface Algorithm. Computer Science Dept., University of Utah, UTECH-CSC-70-101, June 1970.

4. Etkin, B., Dynamics of Atmospheric Flight, Wiley, New York, 1972.

5. Newman, W. M., and Sproull, R. F., Principles of Interactive Computer Graphics. McGraw-Hill, Kogakusha, Tokyo, 1973.

6. Sutherland, I. E., *et al.*, A Characterization of Ten Hidden-Surface Algorithms. Computing Surveys, Vol. 6, No. 1, March 1974.

7. Schumacker, R. A., *et al.*, Study for Applying Computer-Generated Images to Visual Simulations. AFHRL-TR-69-74, U.S. Air Force Human Resources Laboratory, Sept. 1969.

8. Clark, J. H., Hierarchical Geometric Models for Visible Surface Algorithms. Comm. ACM, Vol. 19, No. 10, Oct. 1976, pp. 547-554.

9. Sale, A. H., and Bromley, A. G., A Real-Time Three-Dimensional Graphics Display. Aust. Computer J. Vol. 7, No. 1, March 1975, pp. 15-20.

10. Wylie, C., *et al.*, Half-Tone Perspective Drawings by Computer. Proc. AFIPS, F JCC 1967, Vol. 31, p. 49.

11. Brierly, F., *et al.*, Raster Graphics Image Transformer. RADC-TR-75-175, U.S. Air Force Rome Air Development Centre, July 1975.

12. Knuth, D. E., The Art of Computer Programming, Vol. 3, 'Sorting and Searching'. Addison-Wesley, Mass., 1973.

13. Phong, B. T., Illumination for Computer Generated Pictures. Comm. ACM, Vol. 18, No. 6, June 1975, pp. 311-317.

14. Nelson, D., and Ritchie, M., Using Computer-Generated Displays for Research on Synthesized Displays: Distance Perception Aided by Aerial Perspective and Texture. AMRL-TR-76-34, U.S. Air Force Aerospace Medical Research Laboratory, Aug. 1976.

15. Gouraud, H., Continuous Shading of Curved Surfaces. IEEE Trans. Computers, Vol. C-20, No. 6, June 1971, pp. 623-629.

16. Blinn, J. F., and Newell, M. E., Texture and Reflection in Computer Generated Images. Comm. ACM, Vol. 19, No. 10, Oct. 1976, pp. 542-547.

17. McCleary, L. E., *et al.*, Movie-NUC: A Computer Program to Display Two- and Three-Dimensional Models as Line Drawing or Continuous-Tone Color Images. NUC TP 501, U.S. Navy, Naval Undersea Centre, Dec. 1975.

18. Sutherland, I. E., and Hodgman, G. W., Re-entrant Polygon Clipping. Comm. ACM, Vol. 17, No. 1, Jan. 1974, pp. 32-42.

19. Rife, R. W., Level of Detail Control Considerations for CIG Systems. Proc. 1977 Image Conf., U.S. Air Force Human Resources Laboratory, May 1977, pp. 143-159.

20. Schumacker, R. A., and Rougelot, R. S., Image Quality: A Comparison of Night/Dusk and Day/Night CGI Systems. Proc. 1977 Image Conf., U.S. Air Force Human Resources Laboratory, May 1977, pp. 243–255.

21. Warnock, J. E., A Hidden Surface Algorithm for Computer Generated Halftone Pictures. Computer Sc. Dept., Univ. of Utah, TR 4-15, June 1969.

22. Romney, G. W., Computer Assisted Assembly and Rendering of Solids. RADC-TR-69-365, U.S. Air Force Rome Air Development Centre, Sept. 1969.

23. Bunker, W. M., Computer Generated Images Simulate Infrared and Low Light Level Television Displays. Proc. AIAA Visual and Motion Simulation Conf., April 1977, pp. 120–132.

24. Crow, F. C., The Aliasing Problem in Computer-Synthesized Shaded Images. Comp. Sc. Dept. Univ. of Utah, UTEC-CSC-76-015, March 1976.

25. Mahl, R., Visible Surface Algorithm for Quadric Patches. IEEE Trans. Computers, Vol. C-21, No. 1, Jan. 1972, pp. 1–4.

26. Lobb, D. R., Scanned Laser Visual System Feasibility Study. Tech. Report PMTRADE-TR-76-0001, American Airlines Inc.

27. Loutrel, P. P., A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra. IEEE Trans. Computers, Vol. C-19, No. 3, March 1970, pp. 205–213.

DISTRIBUTION

	Copy No.
AUSTRALIA	
Department of Defence	
Central Office	
Chief Defence Scientist	1
Deputy Chief Defence Scientist	2–3
Superintendent, Science and Technology Programs	4
Australian Defence Scientific and Technical Representative (U.K.)	5
Counsellor, Defence Science (U.S.A.)	6
Defence Library	7
Joint Intelligence Organisation	8
Assistant Secretary, D.I.S.B.	9–24
Aeronautical Research Laboratories	
Chief Superintendent	25
Library	26
Superintendent Systems	27
Divisional File – Systems	28
Author: J. Sandor	29
D. E. Hatton	30
T. Kinsella	31
Materials Research Laboratories	
Library	32
Defence Research Centre, Salisbury	
Library	33
Central Studies Establishment	
Information Centre	34
Engineering Development Establishment	
Library	35
RAN Research Laboratory	
Library	36
Defence Regional Office	
Library	37
Army Office	
Royal Military College Library	38
Air Force Office	
Air Force Scientific Adviser	39
Statutory, State Authorities and Industry	
C.S.I.R.O. National Measurement Laboratory (Chief)	40
C.S.I.R.O. Materials Science Division (Director)	41
Qantas, Library	42
Trans Australia Airlines, Library	43
Ansett Airlines of Australia, Library	44

Applied Engineering Pty. Ltd.	45
Australian Paper Manufacturers (Dr. Norman)	46
BHP Central Research Laboratories, NSW	47
BHP Melbourne Research Laboratories	48
Commonwealth Aircraft Corporation (Manager)	49
Conzinc Riotinto of Australia (Dr. Worner)	50
ICI Australia Ltd., Library	51
H. C. Sleigh Ltd., Technical Department Library	52

Universities and Colleges

Adelaide	Barr Smith Library	53
Australian National	Library	54
Flinders	Library	55
James Cook	Library	56
LaTrobe	Library	57
Melbourne	Engineering Library	58
Monash	Library	59
Newcastle	Library	60
New England	Library	61
Sydney	Engineering Library	62
	Professor R. Wilson	63
N.S.W.	Physical Sciences Library	64
Queensland	Library	65
	Professor A. F. Pillow	66
Tasmania	Engineering Library	67
Western Australia	Library	68
	Dr. P. B. Chapman	69
R.M.I.T.	Library	70

CANADA

International Civil Aviation Organization, Library	71
NRC, National Aeronautical Establishment, Library	72

Universities and Colleges

McGill	Library	73

FRANCE

AGARD, Library	74
ONERA, Library	75
Service de Documentation, Technique de l'Aeronautique	76

GERMANY

ZLDI	77

INDIA

CAARC Co-ordinator Materials	78
Civil Aviation Department (Director)	79
Defence Ministry, Aero Development Establishment, Library	80
Hindustan Aeronautics Ltd., Library	81
Indian Institute of Science, Library	82
Indian Institute of Technology, Library	83
National Aeronautical Laboratory (Director)	84

ISRAEL

Technion—Israel Institute of Technology (Professor J. Singer)	85

JAPAN		
National Aerospace Laboratory, Library		86
Universities		
Tohoku (Sendai)	Library	87
NETHERLANDS		
Central Org. for Applied Science Research TNO, Library		88
National Aerospace Laboratory (NLR), Library		89
NEW ZEALAND		
Air Department, R.N.Z.A.F. Aero Documents Section		90
Universities		
Canterbury	Library	91
SWEDEN		
Aeronautical Research Institute		92
Chalmers Institute of Technology, Library		93
Kungliga Tekniska Hogskolan		94
Research Institute of the Swedish National Defence		95
SWITZERLAND		
Brown Boverie (Management Chairman)		96
UNITED KINGDOM		
Aeronautical Research Council (Secretary)		97
C.A.A.R.C. (Secretary)		98
Royal Aircraft Establishment Library, Farnborough		99
Royal Aircraft Establishment Library, Bedford		100
National Physical Laboratories, Library		101
British Library, Science Reference Library		102
British Library, Lending Division		103
Fulmer Research Institute Ltd. (Research Director)		104
Science Museum Library		105
Universities and Colleges		
Bristol	Library, Engineering Department	106
Cambridge	Library, Engineering Department	107
Nottingham	Library	108
Southampton	Library	109
Strathclyde	Library	110
Cranfield Institute of Technology	Library	111
Imperial College	The Head	112
UNITED STATES OF AMERICA		
NASA Scientific and Technical Information Facility		113
Sandia Group Research Organisation		114
American Institute of Aeronautics and Astronautics		115
The John Crerar Library		116
Allis Chalmers Inc. (Director)		117
Cessna Aircraft Co. (Executive Engineer)		118
Esso Research Laboratories (Director)		119
Texas Instrument Co. (Director)		120
Westinghouse Laboratories (Director)		121
Battelle Memorial Institute, Library		122
Calspan Corporation		123

Universities and Colleges

Brown	Professor R. E. Meyer	124
Iowa State	Dr. G. K. Serory, Mechanical Eng.	125
Polytechnic Institute of New York	Library, Polytech Aero. Lab.	126
California Institute of Technology	Library, Guggenheim Aero. Lab.	127
Spares		128–137